AD-A264 071

TATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final Report 01 Jun 89 - 14 Aug 92

4. TITLE AND SUBTITLE
Application of Error Correcting Codes in fault-tolerant logic-design for VLSI circuits

5. FUNDING NUMBERS
F49620-89-C-0069

6. AUTHOR(S)
Professors P. K. Lala & H. L. Martin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
North Carolina A&T State University
Dept of Electrical Engineering
Greensboro NC 27411

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NE
Bolling AFB DC 20332-6448

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
2305/B1

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

SEE ATTACHED PAGE

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

The use of error detecting/correcting codes in self-checking and fault-tolerant logic design has been receiving considerable attention in recent years. In this report we present the results of our investigation in the application of such codes. We have developed a technique based on low-cost residue code to design arbitrary combinational logic circuits with self-checking capability. We also proposed a technique which allows detection of single, and up to three bits of multi-bit errors in multi-output combinational logic circuits; the major advantage of this technique is that the error detecting capability depends on the output bits of a circuit rather than its internal complexity.

A technique for implementing fully testable sequential circuits from their specifications has also been proposed. This technique eliminates the post-design circuit modifications as used in the currently popular scan-based techniques.

It is now generally accepted that not all faults in VLSI can be implemented by the traditional stuck-at-0/1 fault used at the gate level. To ensure realistic modeling, a faulty transistor in VLSI circuits is modeled as stuck-open or stuck-closed. We have developed a technique to modify CMOS VLSI circuits so that any transistor stuck-open fault in the circuit can be detected using only single test pattern. We also have proposed a new technique for FCMOS and domino CMOS circuits so that they are totally self-checking for all single stuck-open and stuck-closed transistor faults.

Since many faults in VLSI circuits manifest as unidirectional errors at the circuit outputs, a coding scheme for detecting all unidirectional adjacent errors has been developed. This scheme also allows correction of all single bit errors, and up to five bit adjacent unidirectional errors.

We also proposed the constructions of a universal cell, which can function as one of the conventional 2-input gates e.g. AND, OR or as an invertor. The cell is tolerant of all single and certain multiple stuck-closed and stuck-open transistor faults.

The results presented in this report will show that the objectives of the research have been achieved. Future work in this area will be able to use these results to develop new techniques for self-checking and fault tolerant design of VLSI circuits.

Final Report

Application of Error Correcting Codes in Fault-tolerant Logic Design for VLSI Circuits.

Principal Investigator : Dr. P.K.Lala

Co-Principal Investigator : Dr. H.L.Martin

Sponsoring Agency : Air Force Office of Scientific Research

Contractor : North Carolina A&T State University

Contract No. : F49620-89-C-0069

Period of work covered : June 1, 1989 - August 14, 1992

CONTENTS

Abstract

The use of error detecting/correcting codes in self-checking and fault-tolerant logic design has been receiving considerable attention in recent years. In this report we present the results of our investigation in the application of such codes. We have developed a technique based on low-cost residue code to design arbitrary combinational logic circuits with self-checking capability. We also proposed a technique which allows detection of single, and up to three bits of multi-bit errors in multi-output combinational logic circuits; the major advantage of this technique is that the error detecting capability depends on the output bits of a circuit rather than its internal complexity.

A technique for implementing fully testable sequential circuits from their specifications has also been proposed. This technique eliminates the post-design circuit modifications as used in the currently popular scan-based techniques.

It is now generally accepted that not all faults in VLSI can be implemented by the traditional stuck-at-0/1 fault used at the gate level. To ensure realistic modeling, a faulty transistor in VLSI circuits is modeled as stuck-open or stuck-closed. We have developed a technique to modify CMOS VLSI circuits so that any transistor stuck-open fault in the circuit can be detected using only single test pattern. We also have proposed a new technique for FCMOS and domino CMOS circuits so that they are totally self-checking for all single stuck-open and stuck-closed transistor faults.

Since many faults in VLSI circuits manifest as unidirectional errors at the circuit outputs, a coding scheme for detecting all unidirectional adjacent errors has been developed. This scheme also allows correction of all single bit errors, and up to five bit adjacent unidirectional errors.

We also proposed the constructions of a universal cell, which can function as one of the conventional 2-input gates e.g. AND, OR or as an invertor. The cell is tolerant of all single and certain multiple stuck-closed and stuck-open transistor faults.

The results presented in this report will show that the objectives of the research have been achieved. Future work in this area will be able to use these results to develop new techniques for self-checking and fault tolerant design of VLSI circuits.

1. Introduction

The problem of dealing with faults in logic circuits has become increasingly important with the rapidly expanding complexity of VLSI(Very Large Scale Integration) circuits. Three approaches have been adopted to cope with these problems:

i) **Development of sophisticated test generation schemes:**

Significant progress has been made in this area. Several automatic test generation algorithms, which can generate tests for unpartitioned combinational logic circuits comprising of more than 50k gates have been proposed. However, test generation process cannot eliminate the root case of the problem e.g. the improvement of controllability and observability of complex logic circuits.

ii) **Design for Testability:**

The objective of this approach is to apply certain design rules and/or add additional hardware to the original design so that it can be easily tested. However such techniques result in the increase in the physical size and reduce the yield. Furthermore, additional delay is introduced thereby reducing the overall performance of the circuit. In general, design for testability techniques result in significant reduction in test generation costs measured in terms of reduced test generation and fault simulation time.

iii) **Built-in-self-test:**

The major problem with popular design for testability techniques e.g. scan path, LSSD etc is that the circuits designed using these techniques cannot be tested at normal speed and also require long test times. In an attempt to overcome these limitations, another approach known as BIST (Built-in-self-test) has been developed. This approach allows generation of test vectors inside a chip and also has the facility to compress the responses to test patterns into a unique signature. The mismatch in the expected and the actual signatures indicate the presence of a fault in the circuit.

The major disadvantage of the above mentioned approach is that they are unable to cope with transient/intermittent faults, which are emerging as the dominant failure mode in

VLSI circuits. Current test strategies are incapable of coping with this type of fault since these test techniques whether used off-line or implemented as a BIST scheme, are based on the assumption that only permanent faults, resulting from defects in manufacturing process, occur in circuits. The characteristics of intermittent/ transient fault require a test strategy which continuously monitors the operation of a circuit so that if a fault occurs it can be detected during the normal operation of the circuit. This is known as **on-line** or **concurrent** checking.

A basic approach to concurrent checking of logical operation is to utilize fully duplicated logic circuits and compare their outputs. However, while the effectiveness of the technique is well-understood, it requires a large increase in the complexity of the circuitry and therefore, the area on a chip. An alternative strategy is to use coding techniques for concurrent checking of logical operation in VLSI chips.

The basic idea of error-detecting codes is to add extra bits to information bits so that if errors occur they can be detected. These additional bits are called **check bits**. A good error-detecting code will allow multi-bit errors to be detected in information bits. If the check bits allow unique identification of the bits in errors and the original information bits can be reconstructed, the error-detecting code is then not only capable of error detection but also error correction. The cost-effectiveness of any error-detecting code requires that the detection of the given number of errors be achieved with the minimum number of check bits. The cost of error detection depends upon the complexity of the checking hardware, and the time required by checking logic to determine whether any error has occurred.

A wide variety of codes are available for possible use in concurrent checking. For example, parity codes are suitable for random error detection. It is a code that uses a single check bit called a parity bit. The parity bit is determined by the oddness or eveness of the number of 1s contained in the information bits. The advantage of using a simple parity check is that the information bits in a codeword can be processed without decoding. However, it can detect only single bit errors and cannot correct errors. If a double bit error occurs in information bits, the parity is unchanged leaving the error undetected.

Another error-detecting code is the m-out-of-n code, in which all valid codewords have exactly m 1s and (n-m) 0s. Such codes are non-separable code because information is embedded in the codeword with redundancy, and cannot be obtained without using a special decoder circuit. The m-out-of-n codes are useful because of their ability to detect unidirectional multiple errors. Unidirectional errors occur when all the bits in error in information bits appear to undergo either a 1->0 transition or a 0->1 transition; there is evidence that in VLSI chips short circuit faults, power supply line failure etc can cause unidirectional errors.

Another type of error detecting code is the Berger code. Such a code consists of I information bits and C check bits, where

$$C = \lceil \log_2(I+1) \rceil$$

Thus, the total number of bits in an encoded word is I + C. The complement of each bit of the binary representation of the number of 1s in the information bits constitute the check bits. For example, if the information bits are 1011010, check bits will be 011. Hence the Berger code of 1011010 is 1011010011.

The residue code is another type of separable error detecting code. The residue representation of an integer N can be written as

$$N = Im + r \quad r \geq 0$$

where m is the check base, I is an integer. Thus,

$$r = N \bmod m$$

In residue codes the information bits are considered as the binary representation of an integer N. The check bits also constitute a number C. The length of the check bits is $\lceil \log_2 m \rceil$. One type of residue codes known as **low cost residue code** has found applications in on-line error checking in arithmetic operations. A residue code is called a low-cost residue code if the modulus m can be written as

$$m = 2^p - 1$$

where $p \geq 2$ is the number of check bits. All characteristics of the residue number system also apply to low-cost residue codes.

We have developed several techniques for reliable circuit design using error detecting/correcting codes. These techniques are discussed in the following sections.

2. Self-checking combinational circuit design using low-cost residue code

Low cost residue codes have been applied for on-line error detection in arithmetic circuits and PLAs. Currently no general techniques for designing arbitrary combinational circuits with self-checking capability are available. We have developed such a technique based on low-cost residue code (mod 3). In this technique, two residues of the output patterns(information bits) of a circuit are calculated. One residue is derived from the input pattern of the circuit so that it automatically becomes the correct residue of the output pattern corresponding to the applied input pattern. The other residue is directly calculated from the output pattern. Any mismatch between the two residues indicates the presence of a fault in the circuit. A set of logic design rules has been developed which guarantees that a circuit will produce a wrong residue in the presence of a fault. The self-checking design technique has been presented in a paper at the 1991 IEEE VLSI Test Symposium. A copy of the paper is attached with appendix A.

3. Technique for stuck-open fault detection in CMOS

It is now generally accepted that not all faults in VLSI logic can be implemented by the stuck-at-0 and stuck-at-1 fault models used at the gate level. To ensure realistic modeling, faults should be considered at the transistor level. A faulty transistor can be modeled as stuck-open or stuck-closed.

The major problem with the stuck-open fault is that it forces a combinational circuit to behave as a sequential circuit and hence exhibit memory like property. In other words, the circuit output depends not only on the current inputs but also on the past inputs. Testing for a stuck-open p-transistor(n-transistor) requires presetting the output node to logic 0 (logic 1) via an initializing input pattern and then applying a test pattern that establishes a conducting path to reverse the state of the output node to logic 1(logic 0). In other words, a two pattern test strategy has to be used to detect stuck-open faults. The output node changes only if there is no stuck-open fault in the selected conducting path. We have developed a technique to modify CMOS circuits so that any transistor stuck-

open fault in the circuit can be detected using only single test patterns. Two additional transistors and an additional input line are required for this purpose. A paper describing the technique was presented at the 1991 Asilomar Conference, held at Pacific Grove, CA. A copy of the paper is attached with appendix B.

4. Fault tolerant universal cell:

We have proposed the construction of a universal cell designed from NMOS and PMOS transistors. Such a cell is tolerant of stuck-closed or stuck-open faults, and can function as one of the conventional two input gates e.g. AND,OR or an INVERTER. The cell is also tolerant of multiple stuck-closed faults provided there is no more than a single fault in the same path. Multiple stuck-open faults are masked if there are no faults in one of the two signal paths driving the output. The proposed cell design has been published in the International Journal of Electronics (vol. 72, No. 3, 1992). A copy of the paper is attached with appendix C.

5. Self-checking CMOS circuits for stuck-on and stuck-off faults:

Self-checking can be defined as the ability to verify automatically whether there is any fault in the circuit without applying any external test stimuli. Self-checking circuits are very desirable for highly reliable system design, since all faults from a given set would cause a detectable, erroneous output. We have developed a new technique for designing single stage fully complementary metal oxide semiconductor (FCMOS) and CMOS domino logic circuits so that they are totally self-checking for all single stuck-off and stuck-on faults. The technique involves the encoding of the output of the circuit in an error detecting code. CMOS circuits designed using the technique have two outputs. Two of the combinations (01,10) are considered to be valid codewords. The circuit is augmented such that any stuck-off(stuck-on) fault in the modified circuit produces a non-valid output 11 or 00, thus ensuring automatic fault detection. Two papers describing the technique have been published, one in Proc. IEEE 1992 VLSI Test Symposium and the other in IEEE Journal of Solid-State Circuits (August 1992). Copies of these papers are attached with appendix D.

6.Concurrent checking scheme for single and multibit errors in logic circuits:

With the increase in the complexity and density of VLSI chips, transient/intermittent faults have emerged as the dominant failure modes in VLSI circuits. As mentioned previously existing off-line test strategies cannot detect these types of faults since they have been designed to detect permanent faults. Concurrent checking circuits which require continuous monitoring of circuit outputs, can detect transient/intermittent faults as they appear. We have developed a technique for designing circuits with concurrent checking capability. This technique allows the detection of the following errors in the circuit:

i) all single bit errors

ii) all double bit errors(unidirectional and bi-directional)

iii) all triple bit errors are detected

Although this technique has been applied to circuits with upto 8 bits, it can be extended to higher output circuits by taking higher check base as long as the design rules are satisfied. The major advantage of the proposed technique is that the error detecting capability is based on the output bits of a circuit rather than its internal complexity. As long as a fault in a circuit corrupts an output pattern i.e. produces single or multibit error, the probability of its remaining undetected is very low. A paper describing the technique was presented at the 1992 IEEE VLSI Test Symposium. A copy of the paper is attached with appendix E.

7. State assignment technique for fully testable sequential circuit design:

A technique for implementing fully testable sequential circuits from their specifications e.g. state transition graphs has been proposed. This technique eliminates the post-design modifications as in the currently popular scan based technique. We use m-out-of-n code for representing the valid states of a circuit. If the implementation machine corresponding to a specification machine of P states require n flip-flops, the total number of states

$Q = 2^n$. If an m-out-n code is used to represent the P states, the required number of flip-flops can be determined from the following inequality:

$$\frac{(n-1)!}{m!(n-m-1)!} \geq P \geq \frac{n!}{m!(n-m)!}$$

While assigning codewords to P valid states, those states that have identical output(s) are assigned codewords of distance 2d (d >1). However, the number of states which have same output should be less than or equal to $\lfloor n / 2 \rfloor$. The invalid states (=Q-P) will have don't cares as next states, but their outputs should differ from that all P valid states i.e. they should be assigned a unique output value. In order to maximize the number of codewords in m-out-of-n code the best value of m is $\lfloor n / 2 \rfloor$. At most three additional flip-flops with associated driving logic, and one extra output line are needed to implement sequential circuits of arbitrary number of states. A paper describing the technique has been accepted for presentation at the 1992 International Conference on Microelectronics to be held at Monastrir, Tunisia. A copy of the paper is attached with appendix F.

8. A coding scheme for burst error detection and correction:

Errors in computing systems are caused by faults in the components. A number of coding techniques have been proposed over the years to detect and/or correct errors in memory systems, logic circuits and arithmetic circuits. In general single error correcting/double error detecting Hamming codes have been used in the memory systems, whereas residue codes have been used for error detection in arithmetic circuits.

It has been observed that various faults in VLSI devices manifest as unidirectional errors at the output of a device. Burst errors i.e failure in adjacent bits are also common in memory systems. Two important classes of all unidirectional error detecting codes (AUED) are Berger code and m-out-of-n code.

We have developed a new coding scheme for detecting all unidirectional adjacent errors. The scheme also allows correction of all single bit errors, and upto five bit adjacent unidirectional errors.

The following definitions will later be used to construct the proposed code:

Definition 1:

The *residue weight* of an information bit is the residue of the binary weight of the bit in mod m, where m is the selected check base. If mod 13 is selected as the check base, there are 12 residue weights.

1 2 4 8 3 6 12 11 9 5 10 7

Definition 2:

The *residue complement* (rc) of a valid residue weight r is rc = m-r, where m is the check base.

For example if mod13 is selected as the check base, the residue weights from 7 to 12 are complements of 6 to 1 respectively.

Definition 3:

Two residues are said to be *unique* if they are not the same or complement to each other. For example two residue weights a and b are unique if and only if a = b = m - b where m is the check base.

Example: mod13 has 6 unique weights from 1 to 6.

Definition 4:

A *group* can consist of at most four elements, and should be constructed in such a way that the residue of the sum of any two adjacent elements should be greater than zero. Also the residue of the sum of all four elements in the group should not be equal to zero.

Definition 5:

Two groups are *adjacent* to each other if and only if the residue of the sum of last two elements in the first group and the first two elements in the other group is not equal to zero

Coding Scheme:

We assume that the number of information bits k is equal to 2^i (where i=3,4,5..), and the check base is m ($=2^{i+1}$ -3). The no of unique residue weights for such a check base m is [m/2]. Table 1 shows the check base and the corresponding unique residue weights for various lengths of information bits.

Information bits	Check base	No. of unique residue weights
8	13	6
16	29	14
32	61	30
64	125	62

Table 1: Number of unique residue weights corresponding to the check base.

In this section we propose a new coding scheme which has k information bits and c check bits. The c check bits consist of two fields R and X. The R-field consists of ($\log_2 k+2$) and the X-field has [$5.2^{\log_2 k-4}$] bits (see Appendix G). Table 2 shows the number of check bits in R and X fields against the number of information bits. In the R-field ($\log_2 k+1$) bits are used to represent the residue of the information bits, the remaining bit represents the parity over the ($\log_2 k+1$) bits or a subset of these bits. Table3 shows how this parity bit is calculated for words of varying information bits. For example, in case of 8-information bits 13 is the check base. Four bits b3,b2,b1 and b0 are needed to represent the residue of the information bits. The parity bit i.e. the fifth bit in the R-field is obtained by ex-oring the bits b2 and b3.

Information bits	Check bits		Total Check bits
	R field	X field	
8	5	3	8
16	6	5	11
32	7	10	17
64	8	20	28
128	9	40	49

Table 2: Number of information bits and the corresponding check bits

Information bits	R-field	
	No of bits needed for residue representation	parity bit
8	b3 b2 b1 b0	over b3 & b2
16	b4 b3 b2 b1 b0	over all bits
32	b5 b4 b3 b2 b1 b0	over b1,b2,b4 & b5
64	b6 b5 b4 b3 b2 b1 b0	over all bits

Table 3 : Generation of Parity bit in the R-field

The code construction is based on the following lemmas:

Lemma 1:

All errors in the information bits will be detected if the check bits regenerated from the corrupted data are different from the original check bits.

Proof:

Let X be the original data and Y be the derived check bits. Let Z be the modified data due to change in one or more bits in the original data and Y' be the derived new check bits. If Y xor Y'= 0, the error cannot be detected. Thus in order to detect all errors, Y must be different from Y'.

Lemma 2:

An error(s) in the information bits can be corrected if the check bits in the presence of this error, is different from the check bits produced due to any other error in the information bits.

Proof:

Let X be the information bits and Y be the check value. Let Y1,Y2...YN be the check bits due to the presence of different combinations of errors in the information bits. Since Y1, Y2 and YN are unique, the syndrome patterns for these errors will be unique, hence all these errors will be corrected.

Lemma 3:

If the residue of certain information bits is R, and the residues R0 and R1 for error patterns e0 and e1 respectively, are complementary, then e0 and e1 can be corrected if they have a common bit.

Proof:

Assume two - multibit error patterns in the information bits x0x1x2....xn, which have atleast one bit in common. For example, let us assume two possible unidirectional error patterns e0 and e1 with x0 bit being common:

$$e0 = x0\ x1\ x3\ x5$$

$$e1 = x0\ x2\ x4\ x6$$

The residue for e0 and e1 are R0 and R1 respectively:

$$R0 = (r0+r1+r3+r4) \bmod m$$

$$R1 = (r0+r2+r4+r6) \bmod m$$

where r0, r1, r2, r3, r4, r5 and r6 are residue weights of bits x0, x1, x2, x3, x4, x5 and x6 respectively.

Since by assumption, R0 and R1 are complement to each other, R0+R1 = m.

If one of the two error patterns occur in the information bits, then the erroneous residue R' can be one of the following:

case i:

$$R' = R+ R0 \text{ or } R' = R- R1$$

case ii:

$$R' = R+ R1 \text{ or } R' = R- R0$$

Since the values of R and R' are known, the original residue can be reconstructed by adding R0 or substracting R1 from R' in case i, and adding R1 and substracting R0 from R' in case ii. The common bit x0 will be either 1 or 0 in the presence of one of the error patterns. If it is 1, we can substract R0 or R1 from the erroneous residue, i.e. R,' and if x0 is 0, we add R0 or R1 to R'. Therefore based on the status of x0 bit the erroneous patterns can be uniquely identified as shown below:

case i: If R = R' - R0 or R = R'+R1, and

=0 - x0x1x3x5 are erroneous

common bit x0

=1 - x0x2x4x6 are erroneous

case ii: If R = R' - R1 or R = R'+R0, and

=0 - x0x2x4x6 are erroneous

common bit x0

=1 - x0x1x3x5 are erroneous

Hence the error patterns e0 and e1 are correctable. QED.

Code Construction:

The proposed code is constructed using the following steps:

(i) Assign k residue weights corresponding to the selected check base, m, to the k information bits. These weights are obtained by calculating the residues of the decimal weights 2^0 to 2^{k-1}.

(ii)The residue weights corresponding to the k information bits are partitioned into 5.2^{i-4} groups such that each group has a minimum of 2 elements and a maximum of 4 elements.

(iii) Elements in a group, G, can be selected in the following manner

case a:

If two elements a & b are placed in a group (a b),

select any two elements from the k-residue weights for a & b.

case b:

If three elements a,b & c are placed in a group (a b c),

select any residue weights for a & b. The residue weight for the third element c should be selected such that

residue (a+b) = m - residue (b+c)

case c:

If four elements a, b, c & d are placed in a group (a b c d),

select a b c d such that they will satisfy one of the following conditions

residue (a+b) = m - residue (b+c)

residue (b+c) = m - residue (c+d)

residue (a+b) = m - residue (a+b+c+d)

residue (b+c) = m- residue (a+b+c+d)

residue (c+d) = m - residue (a+b+c+d)

Form as many groups as possible which satisfy the above cases, provided the same residue weight is not present in more than one group.

(iv) Order the groups such that

a) the neighboring groups are adjacent

b) the residue of the sum of two consecutive bits in a group should not be the same as the residue of the sum of two consecutive bits in any other group.

c) all residues corresponding to the sum of the last two bits in one group and the first two bits in its adjacent group should be different and should not be equal to the residue of the sum of two consecutive bits obtained in substep (b).

d) two residues obtained in substeps b and c can be complementary to each other if they share at least one common bit.

Example: Consider the grouping (a b c) (d e f) (g h)

substeps b & d: the residues (a+b) mod m , (d+e) mod m and (g+h) mod m should be unique, whereas the following residues can be complement to each other:

(a+b) mod m and (b+c) mod m

(d+e) mod m and (e+h) mod m

substeps c & d: the residues (b+c+d+e) mod m and (g+h) mod m should be unique. Residues (b+c+d+e) mod m and (e+f+g+h) mod m can be complement to each other.

(v) Determine the parity of each group, it represents a bit of the X-field.

(vi) Determine the R-field.

Let us illustrate the code construction procedure by considering k= 8 and check base of mod 13.

bits	d7	d6	d5	d4	d3	d2	d1	d0
binary weights	128	64	32	16	8	4	2	1
residue weights(mod13)	11	12	6	3	8	4	2	1

Steps i-iii: one possible grouping is

G1	G2	G3
(2 6)	(11 12 4)	(3 1 8)

Step iv: a possible assignment of residue weights to the information bits is

d7	d6	d5	d4	d3	d2	d1	d0
2	6	11	12	4	3	1	8

Consider an 8-bit binary pattern as shown below

	d7	d6	d5	d4	d3	d2	d1	d0
	1	0	0	1	0	0	1	0
residue weights	2	6	11	12	4	3	1	8

Step v: X-field is derived as follows:

x3 = d7 xor d6 = 1 xor 0 = 1.

x2 = d5 xor d4 xor d3 = 0 xor 1 xor 0 = 1.

x1 = d2 xor d1 xor d0 = 0 xor 1 xor 0 = 1.

Therefore X-field is x3 x2 x1 = 1 1 1

Step vi: residue of the information bits= (2+12+1) mod 13 = 2 = $(0\ 0\ 1\ 0)_2$ = (b3 b2 b1 b0)

Parity bit in the R-field = b3 xor b2 = 0 xor 0 = 0.

Therefore R-field is b3 b2 b1 b0 p = 0 0 1 0 0

The check bits are appended to the information bits so that all the unidirectional errors upto 5-bits (including the check bits) are corrected. Check bits are appended to various information bits as shown below.

8- bit information (d7-d0)

(b1 b2 x2 x3 b3)	(d7 d6)	(d5 d4 d3)	(d2 d1 d0)	(x1 p b0)
check bits	group1	group 2	group 3	check bits

	d7	d6	d5	d4	d3	d2	d1	d0
residue weights	2	6	11	12	4	3	1	8

Correction Algorithm:

Let (dk-1.....d1d0) be the information part, x1....x5.2^{logk-4} be the X-field and b0....$b_{|logk|}$ p be the R-field in the received word.

1) Find the xor field, x', from the information bits of the received word, and take the xor of x and x'

$$dx = x \text{ xor } x'$$

2) Find the residue value, r' and the parity bit p', from the information bits. Substract r' from r, and xor p and p'.

$$dr = r - r' \quad \text{and} \quad dp = p \text{ xor } p'$$

3) If dx = 000..0 and / or dr = 0 and / or dp = 0, the presence of an error(s) in the information bits, or in the residue and / or the xor field is implied.

Tables 6 shows the dx, dr and dp values and the corresponding errorneous bits for the (16,8) code.

xor field (dx)	Difference between expected and actual residue(dr)	parity bit (dp)	Erroneous bit(s) (unidirectional)
	8(5)	0	d7d6
	10(3)	0	d5d4
	3(10)	0	d4d3
	4(9)	0	d2d1
	9(4)	0	d1d0
	5(8)	0	d7d6d5d4
0 0 0	7(6)	0	d4d3d2d1

	0	1	p
	4(9)	1	b2
	8(5)	1	b3
	1(12)	0	b0
	2(11)	0	b1
	6(7)	1	b1b2
	1(12)	1	b0p
	2(11)	0	d7
	6(7)	0	d6
100	3(10)	0	d6d5d4
	1(12)	1	x1b0p
	10(3)	1	x1b0pd0d1
	0	1	x1p
	9(4)	0	d0d1x1
	9(4)	1	d0d1x1p
	0	0	x1
	11(2)	0	d5
	12(1)	0	d4
	4(9)	0	d3
010	1(12)	0	d5d4d3
	6(7)	0	d7d6d5

	8(5)	0	d3d2d1
	4(9)	1	b2x2
	6(7)	1	b1b2x2
	0	0	x2
	3(10)	0	d2
	1(12)	0	d1
	8(5)	0	d0
001	12(1)	0	d2d1d0
	6(7)	0	d4d3d2
	8(5)	1	x3b3
	3(10)	1	x3b3b7b6
	0	0	x3
	4(9)	0	d6d5
110	7(5)	0	d6d5d4d3
	7(5)	0	d3d2
	3(10)	0	d3d2d1d0
	0	0	x2x3
	4(9)	1	x2x3b2
	8(5)	1	x2x3b3
011	6(7)	1	x2x3b2b1
	12(1)	0	x2x3b2b3

	1(12)	0	x2x3b2b1b3
	3(10)	1	x2x3b3d7d6
	8(5)	0	x1d0
	12(1)	0	x1d2d1d0
101	12(1)	1	x1d2d1d0p
	8(5)	1	x1d0p
	9(4)	1	x1d0pb0
	10(3)	1	x3b3d7
	10(3)	0	d6d5d4d3d2
111	10(3)	1	x2x3b3d7
	1(12)	0	b2x2x3b3d7

Table 4: dx, dr, dp fields and the corresponding erroneous bits for (16,8) code.

9. Conclusion:

In this project we have investigated the use of error-detecting codes in self-checking logic design. We also proposed the design of a universal fault-tolerant cell. In addition a coding technique for detecting and correcting unidirectional burst errors has also been proposed. So far the work has resulted in two journal papers and five refereed conference papers. Five students have completed their MS theses during the three year project period. The titles of these theses are given below:

1. Application of residue code in automatic error detection and correction in logic circuits. (F.Busaba, 1990)

2. Self-checking and self-correcting design using separable codes. (Z.Xie, 1991)

3. Novel coding techniques for single and multi bit error detection and correction. (K.C.Yarlagadda, 1991)

4. Testable and self-checking design of CMOS circuits. (M.S. Cheema, 1992)

5. State assignment techniques for fully testable and totally self-checking sequential circuits. (X.Ma, 1992)

Research in this area has resulted in the identification of two topics which need further investigation:

1. Enhancement of off-line testing using checkers
2. A design methodology for self-checking CMOS circuit design.

In a self-checking circuit the output is a codeword in the absence of a fault from a given set. The checker circuit produces a non-codeword if the output of the functional circuit is a non-codeword or the checker circuit itself has a fault. Although a checker is included specifically for on-line monitoring of the functional circuit output, it can also be used during off-line testing. If the output is not a codeword, then the circuit can be considered to have a fault and further testing is not necessary. On the other hand, if the outputs of a functional circuit are all codewords, the absence of a fault cannot be guaranteed; this may happen if a circuit has more than one fault and the circuit is not fault secure in the presence of multiple faults. Thus, conventional off-line testing will still be necessary to detect the presence of faults. The focus of the research should be how to augment the functional circuit such that the output values, in response to test inputs, can be compressed into a unique pattern. After the pre-determined input patterns have been applied to the functional circuit, the compressed output response can be shifted out for comparison with a reference value. Thus, even if the functional circuit produces correct codewords in presence of a fault or a set of faults, the compressed output response will indicate the presence of the fault.

As far as we are aware, no general methodology is available for self-checking CMOS circuits. The focus of our research will be to extend the concept of self-testing and

fault-secureness to both static and dynamic CMOS circuits. We have shown that self-checking at the transistor level is feasible. However, so far we have considered only single-output complex CMOS gates. Our next objective is to study techniques of augmenting sequential circuits implemented in CMOS so that they will be totally self-checking for the important class of CMOS defects i.e. breaks and stuck-on transistors. One important consideration will be to use minimum number of additional transistors and/or input lines.

Appendix A

An approach for designing self-checking logic using residue codes

P.K.Lala, F.Busaba, K.C.Yarlagadda
Department of Electrical Engineering
N.C. A&T State University, Greensboro.N.C.27411, U.S.A

Abstract:

It is generally agreed now that the major portion of faults in logic system are not of permanent nature. Current testing strategies are incapable of detecting non-permanent faults. The characteristics of such faults requires that logic circuits be designed in a way so that if there is a fault in the circuit, its effect will be detected during the normal operation of the circuit i.e. the circuits be self-checking. In this paper we propose two rules based on the mod 3 residue coding scheme for designing circuits for on-line error detection.

1.Introduction

Concurrent or on - line testing of VLSI circuits are becoming increasingly popular because of the limitation of the current built - in - test schemes and off - line testing. Periodic off -line testing may be used to ensure detection of permanent faults, but is not effective against temporary i.e. transient / intermittent faults, which are emerging as the dominant failure mode in VLSI circuits [1]. On-line testing, on the other hand, enables the detection of error(s) due to permanent as well as temporary faults. In general, on - line - testing is possible if only a circuit has been designed in as way so that it can determine during its normal operation whether it contains a fault or not i.e. if the circuit is self-checking

Self-checking circuits are typically designed using coding techniques [2]. The basic approach is to design the circuit so that it produces a valid codeword in the absence of a fault; in the presence of a fault the output pattern contains erroneous bit(s), thus producing a non - valid codeword.

Several error detecting codes e.g. parity code, Berger code, m - out - of n code and residue codes have been considered for application in design for on -line testing. In this paper we present a new approach for designing combinational logic circuits using residue codes. Traditionally, residue codes have been used for checking arithmetic operations such as addition,multiplication etc. Such codes allow the derivation of check bits of an operation directly from the operands. It has been demonstrated that low - cost residue codes can be incorporated into VLSI circuits for on -line detection of permanent and transient errors [3], however, so far the application of residue codes have been restricted to common types of devices e.g. multipliers, ALU etc. As far as the authors are aware, no work has been reported so far to find design rules for implementing circuits so that the residue of an output pattern of a circuit in the presence of a fault(s) will be different from that of the fault - free output pattern. This is extremely important, otherwise many faults will be masked because an output

1991 IEEE VLSI Test Symposium

pattern in the presence of a fault may be different form the fault - free response, but the corresponding residues could be the same. In this paper, we present two design rules which will eliminate this problem in combinational logic circuits, thus making the circuits self-checking for single bit error(s).

2.Overview of self - checking design using mod 3 residue code

The residue number system and its properties are well documented [4]. In this section we briefly discuss the properties of low - cost residue codes. A residue code is called a low - cost residue code with check base or modulus m if and only if $m = 2^{p-1}$, $p \geq 2$, where p is the number of check bits. In the paper, we would assume m = 3. The main features of mod 3 residue code are as follows:

i. The check bits in a codeword are only two bits.

ii. The residue of 2^i(mod 3) is 1 if i is even and 2 if i is odd.

Fig. 1 illustrates the basic scheme for using mod 3 residue code in on - line testing of arbitrary combinational circuits. The residue generator 1 is used to generate the residue of an output pattern from the input pattern producing this output. The residue generator 2 generates the complement of the residue directly from the output pattern. A two rail checker compares the two residues and produces a non - codeword i.e. 00 or 11 if the two residues are not complements of each other.

In order to understand how mod 3 residue code can be used in error detection, let us consider a combinational circuit, with integers Z and Z' representing the fault - free and faulty output pattern respectively corresponding to an input pattern, then $Z' - Z = \pm w$

If w mod 3 ≠ 0, a single error or a multi bit error is associated to be present in the output pattern. However, w mod 3 = 0 does not necessarily mean that an output pattern is error free; this is because w could be a multiple of 3, even though Z' and Z are not equal, thus masking the presence of error(s) in the output pattern. To illustrate this, let us assume that the fault - free output of a combinational circuit for a particular input pattern is,

$$f3\ f2\ f1\ f0 = 1\ 0\ 0\ 0 = 8$$

Suppose the presence of a fault in the circuit, affects the output pattern so that

$$f3\ f2\ f1\ f0 = 1\ 0\ 1\ 1 = 11$$

Thus, w = + 3 (= 11 - 8) = 0 (mod 3), hence the fault cannot be detected. However, by proper "distribution" of the outputs, the fault can be detected. The "distribution" corresponds to changing the position of certain bit(s)

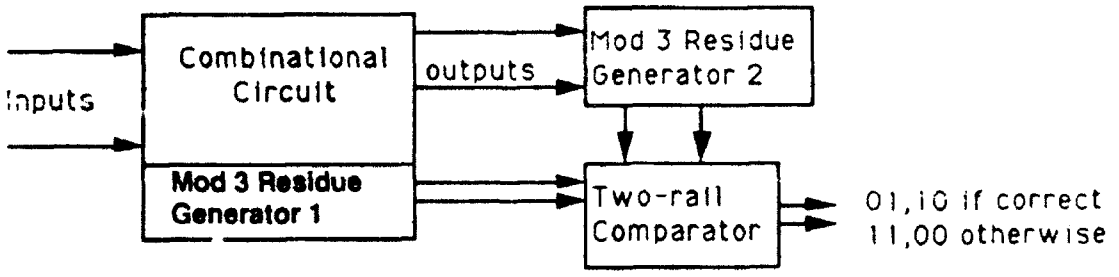

Fig 1. Block diagram of self-checking circuit

hence the weight(s), in the output pattern. For example, if we distribute the output bits as

f3 f1 f2 f 0 = 1 0 0 0 = 8 = 2 mod 3

the same assumed fault will change the output pattern to

f3 f1 f2 f 0 = 1 1 0 1 = 13 = 1 mod 3.

As can be seen , the distribution of the output bits has resulted in different residue, indicating the presence of a fault.

The above example is straight forward in the sense the distribution of the outputs can be done intuitively; for circuits with more than four outputs, the problem becomes more complicated. In the next section, we present two design rules for eliminating this problem, so that any single error or unidirectional multi bit error can be detected on - line.

3.Rules for self-checking design

Before presenting the rules, we need to define certain terms.

Definition : A set of outputs, F, form a cluster if there exists a single fault that can be propagated to all elements of the set at the same time.

Definition : A cluster that has i elements is denoted by Gi.

For example, the outputs f0, f1 and f2 in Fig 2 can be in one cluster G3, since a stuck-at-1 fault at A is propagated to all of them. Similarly, f2 and f3 form a cluster G2 because a stuck-at-1 fault at D is propagated to both outputs. Note that G2 is a sub-cluster of G3

Example 1:

As can be seen in Fig. 2, a fault at D can propagate to f2 and f3. Suppose the fault-free output is:

f3 f2 f1 f0 = 0011=0 (mod 3)

If D is stuck-at-1, the output will be

f3 f2 f1 f0 = 1111=0 (mod 3.)

The residue of the output does not change because cluster G2=(f2 f3) adds 3, to the original residue. This is because starting from bit 0, an even bit contributes 0 and

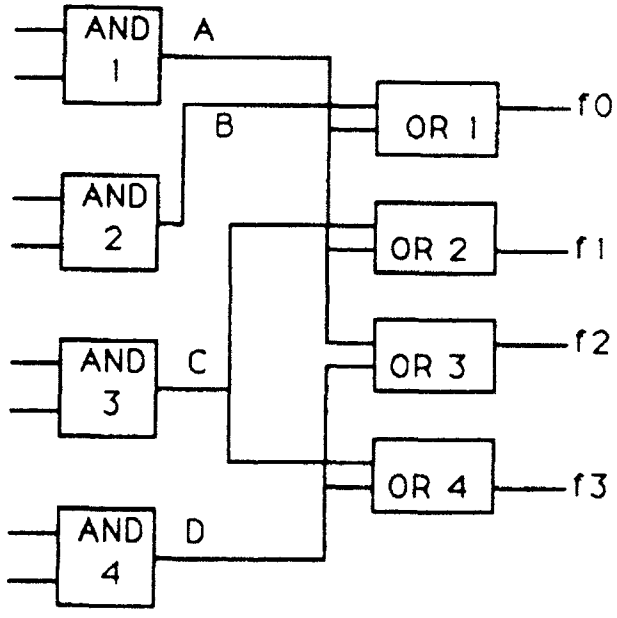

Fig. 2 Combinational Circuit

an odd bit contributes 2 to the residue. Thus, bits f0,f1,f2 and f3 contribute 1,2,1 and 2 respectively to the residue. The following lemmas suggest how the output can be distributed to avoid masking of fault(s).

Lemma 1

The elements of a cluster Gi have to be distributed in such a way so that the residue of the output pattern will not be equal to a number which is a multiple of 3, in the presence of a fault.

Proof: Let us assume that a fault create either a single bit error or a unidirectional multi-bit error at the output. Let r be the residue of an output pattern in the fault-free circuit. If a fault affects a certain cluster G, the output bits in the cluster will either change to all 1s or to all 0s, thus adding to or subtracting from r a certain number l. The new residue, r', is

$r' = r \pm l \bmod 3$.

If l is a multiple of 3, l mod 3 will be zero, leaving r unchanged. Thus, the outputs have to be distributed in order to make l mod 3 ≠ 0, which in turn makes r≠r'. Q.E.D.

Lemma 2:

Cluster G1 i.e a cluster with a single element i will produce wrong residue if bit i is erroneous.

Proof: If i is erroneous the new residue r' is

$r' = r \pm \bmod 3\ (2^i)$

Depending on the position of i ie odd or even, the residue of 2^i will be either 2 or 1. Consequently, r'≠r. Q.E.D.

The following rule is based on the above lemmas.

Rule 1

Cluster G2 should have both of its elements located at either odd bit locations or at even bit locations in order to produce wrong residue in the presence of a single fault.

The rationale behind this rule is as follows

Since the fault affects two output bits, the new residue,r', will be the addition (or subtraction) of a number l to the original residue, r. Two odds bits or two even bits will add to or subtract from the original residue 1(=(2+2) mod 3) or 2 (=(1+1) mod 3) respectively. Thus r' will be different from r. On the other hand, if one of the bits is odd and the other is even, then l =0(= (1+2) mod 3), thus r = r', and hence the fault will not be detected.

Let us illustrate the use of the rule. In Fig 2, we note that cluster G2 = (f2 f3) has its elements distributed as one even and one odd. Thus fault D s-a-1 does not change the residue. If, however, f2 and f3 are redistributed in the following order, f3 f1 f2 f0, (i.e. f0,f2,f1,f3 have weights of 1,2,4,8) the residue for the error-free output pattern, and for the pattern in presence of the fault D s-a-1 will be 2 and 0 respectively. Thus, the fault can be detected.

In multi-output circuits, it is possible for fault to affect more than two outputs i.e. there could be clusters containing three or more elements. To illustrate, let us consider the following possibilities in Fig 2.

1. If B = C = D = 1, the fault is masked because f1, f2, and f0 will be 1 in fault free circuit.
2. If B = 0 and D = C = 1, the fault will only propagate to f0; f1 and f2 will remain unchanged. Thus G1 = (f0).
3. If B = C = 0 and D = 1, the fault will propagate to both f1 and f0, leaving f2 unchanged. Thus G2 = (f0 f1)
4. If B = D = 0 and C = 1, the fault will propagate to f0 and f2; f1 and f3 will remain unchanged. Thus, G2=(f0,f2)
5. If C = D = 0 and B = 1, the fault will propagate to f1 and

f2, fo and f3 will remain unchanged. thus, G2 = (f1, f2).

6. If B = D = C = 0, the fault will propagate to f0, f1 and f2. Thus G3 = (f0 f1, f2).

Thus fault A s-a-1 can generate any one of the following clusters depending on the input pattern applied to the circuit.

(f0), (f0, f1), (f0, f2), (f1,f2),(f0, f1, f2).

It has been shown that the elements of a cluster Gi($i \geq 3$) can be distributed in a way so that a fault which affects all i elements of the cluster, will generate an erroneous residue [5]. However, such a distribution cannot guarantee that all sub-clusters of Gi corresponding to the same fault, will also produce wrong residue. In other words, for a particular input pattern the circuit will produce a wrong residue, whereas for other patterns the residue may or may not be wrong. The following rule alleviates the problem.

Rule 2:

Decompose a cluster G ($i \geq 3$) by incorporating additional gates, so that the output of each new gate cannot affect more than two outputs. In other words, a fault will create clusters of not more than two elements

Fig. 3 shows how an additional AND gate , A , is incorporated in Fig. 3, so that G3=(f0, f1, f2) can be decomposed into G1 = (f0) and G2 = (f1,f2). In this augmented circuit, single fault can create one of the following clusters:

(f0), (f1,f2), (f1,f3), (f2,f3)

In order to ensure that a fault will produce an erroneous residue, the output bits need to be distributed as follows

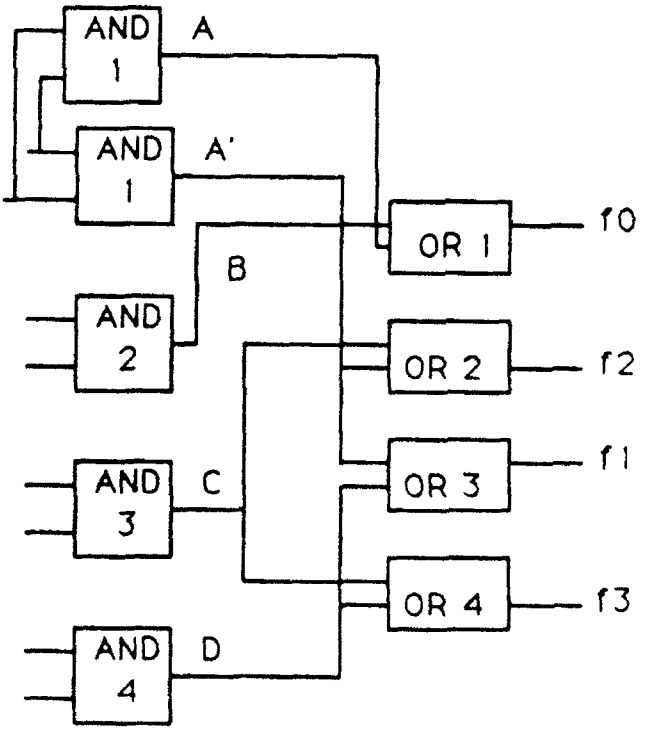

Fig. 3 Augmented version of Fig 2

using Rule1.

even	odd	even	odd	even
f3	-	f2	f0	f1

Note that the above distribution requires that the residue generator 2 in circuit of Fig. 1 assign a weight of 16 to bit f3.

4.Conclusion

Two rules for designing self-checking circuits based on mod 3 residue code have been proposed. These rules are applicable to arbitrary combinational circuits. The additional circuitry needed by this design approach is offset the advantage that the circuit would be continuously monitored during its normal operation, thus allowing the detection of faults as they occur. Although two-level AND-OR logic is assumed in the paper for illustration, the rule are also applicable to multi-level logic; however, in the case of multi-level logic the formation fo clusters may not be as straightforward as in the two-level logic. Because PLAs are basically AND-OR structures, the proposed rules could also be used to design self-checking PLAs.

5. *References:*

1. T.R. Rao and H.Fujiwala "Error Control Coding for Computer Systems", Prentice Hall,1989.

2. I.L. Sayers and D.J. Kinniment "Low-cost residue codes and their application to self-checking VLSI systems" IEE Proc., Vol 132, No.4, July 1985, pp. 197-202.

3.N.S. Szabo and R.L. T auaka "Residue Arithmetic and its applications to Computer Technology " Mc Graw-Hill, 1967.

4.J.H. Lala and L. Alger "Hardware and software fault tolerance a unified architectural approach" Proc Symp on Fault Tolerant Computing, 1988, pp 240-245

5.F.BuSaba "Application of residue code in automatic error detection and correction in logic circuits " MS Thesis, Dept. of Elect .Engg., North Carolina A&T State Univ., 1990.

Acknowledgement:

This work was supported by the Air Force Office of Scientific Research under grant F49620-89-C-0069.

Appendix B

Scheme for detecting CMOS stuck-open faults using single test patterns

Manjit S. Cheema
Department of Electrical Engineering
N.C. A&T State University,
Greensboro.N.C.27411.

P. K. Lala
Department of Electrical Engineering
N.C. A&T State University,
Greensboro.N.C.27411.

Abstract

It has been widely accepted that not all defects in CMOS logic can be represented by the stuck-at fault model. One example is the transistor stuck-open fault. The major problem with stuck-open faults is that it forces a combinational circuit to behave as a sequential circuit. It has been established that two pattern test technique may be inadequate to detect all stuck-open faults in the presence of stray ciruits delays and glitches. Furthermore 'robust tests' that can detect transistor stuck-open faults independent of glitches and circuit delays may not exist for all stuck-open faults. This paper presents a technique to modify CMOS circuits so that any transistor stuck-open fault in the circuit can be detected using only single test pattern. Two additional transistors and an additional input line are required for this purpose. It is also necessary to incorporate an inverter at the final output of the circuit.

1. INTRODUCTION

Complementry MOS (CMOS) technology has emerged as the the dominant technology for manufacturing VLSI digital circuits[1]. A CMOS circuit consists of a P-network connected between Vdd and the output node, and a N-network connected between GND and the output node.The circuit for P-network and N-network bear a dual relationship by DeMorgan's theorem. Fig.-1 shows the block diagram of a CMOS circuit. It is now generally accepted that not all faults in CMOS VLSI logic can be modelled by the stuck-at-0 and stuck-at-1 models used at the gate level. An example is the transistor stuck-open(s-open) fault.

The major problem with the stuck-open fault is that it forces a combinational circuit to behave as a sequential circuit and hence exhibit memory like property[2-4]. In other words, the circuit output depends not only on the current inputs but also on past

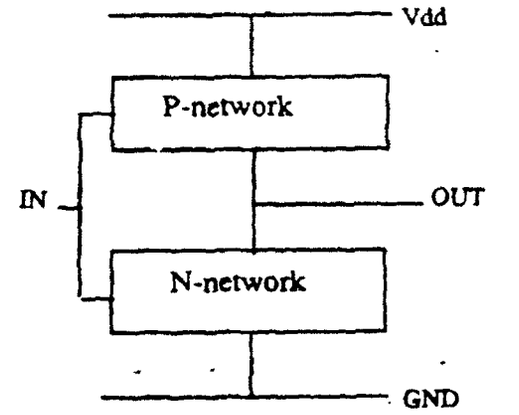

Fig.1 Block diagram of FCMOS circuit.

inputs. Testing for a stuck-open p-transistor (n-transistor) requires presetting the output node to logic 0 (logic 1) via an initializing input pattern and then applying a test pattern that establishes a conducting path to reverse the state of the output node to logic 1 (logic 0). In other words a two pattern strategy has to be used to detect stuck-open faults. The output node changes only if there is no stuck-open fault in the selected conducting path. It has been shown in [5,6] that two pattern tests may be invalidated in static CMOS circuit because of arbitrary circuit delays, glitches and timing skew in input changes. However these problems can be avoided by redesigning CMOS circuits[7-10]. In this paper a new design technique for testable CMOS circuits is presented. This technique makes the circuit fully testable for all single stuck-open faults using only single test pattern. The problem of test invalidation by glitches and circuit delay etc. is totally eliminated.

1058-6393/91 $1.00 © 1991 IEEE

2 Proposed approach for stuck-open fault detection

In order to detect stuck-open faults the the original CMOS circuit has been augmented by using two pass transistors(p1,n1), an inverter(p2,n2) and an external input line 'X'. The inverter acts as a buffer to boost the signal at the ouput. The block diagram implementation of the proposed scheme is shown in Fig.2. It should be noted that only a single inverter will be used to drive the final output of circuit under test, irrespective of the number of stages in the circuit.

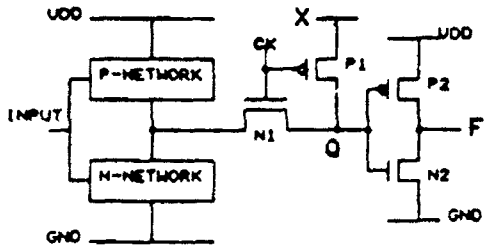

FIG.2 BLOCK DIAGRAM OF AUGMENTED CMOS CIRCUIT

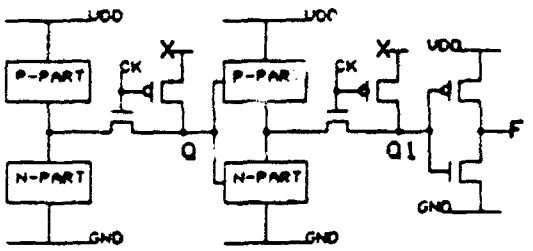

FIG.3 BLOCK DIAGRAM OF AUGMENTED TWO STAGE CMOS CIRCUIT

Hence if a circuit has several stages, the inverter is needed only to drive final output of the circuit, the intermediate stages can drive the following stages directly. This is illustrated in Fig.3. The first stage does not have any inverter at its output. It is assumed that the circuit gives the final output only after Ck makes a transition from 0 to 1. In theorem1 it is proved that any single stuck-open fault in the circuit can be detected by a single test pattern.

THEOREM1: All single stuck-open faults in the P-part or N-part of a CMOS circuit modified as shown in Fig.-2 or Fig.3 can be detected by using only a single test pattern.

Proof: During the test mode when the clock is low, the output of the circuit is always set to the opposite value to that expected, so that for any s-open fault in the selected conducting path the circuit output retains this value. Therefore when the P-network (N-network) is selected for testing and a selected test pattern is applied to the circuit, the output Q is set at logic 0(1). The transmission path from Vdd (GND) to the output node 'Q' will be activated by the test pattern if only if a stuck-open fault is not present in the selected path. Hence the output node Q will change from logic 0 (logic 1) to logic 1(logic 0) iff there is no s-open fault in the selected path in P-network (N-network). In other words if the output Q of the P-network (N-network) remains at logic 0 (logic 1) in response to a selected test pattern, a stuck-open fault has been detected. QED.

Let us apply the proposed technique to the CMOS circuit shown in Fig.4(a).The augmented version of this circuit is shown in Fig.4(b).

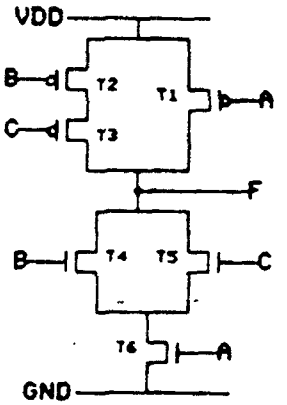

FIG.4(A) CMOS CIRCUIT IMPLEMENTING FUNCTION $F=\overline{A(B+C)}$

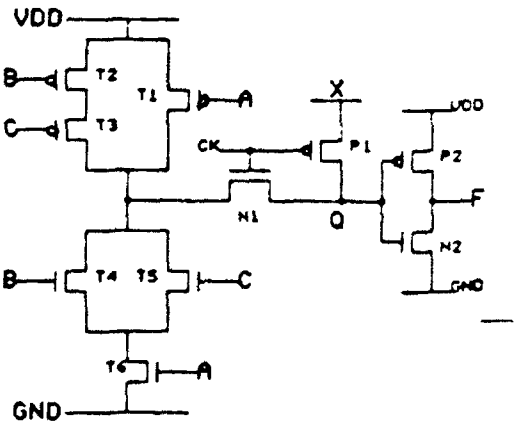

FIG.4(B) AUGMENTED CIRCUIT FOR STUCK-OPEN FAULT DETECTION

Let us suppose transistor T1 is stuck-open. The test pattern for this fault is : A=0, B=1, C=0, X=0. When the test pattern is applied, output node 'Q' will be set to logic '1' iff T1 is not stuck-open else it will be set to logic '0', hence the fault will be detected. A test pattern for each transistor stuck-open fault in the circuit of Fig.4(B) is given below:

Fault	Test pattern	Faulty(F)	Fault free(F)
T1 s-open	A=0, B=1, C=0, X=0	1	0
T2 or T3 s-open	A=1, B=0, C=0, X=0	1	0
T4 or T6 s-open	A=1, B=1, C=0, X=1	0	1
T5 or T6 s-open	A=1, B=0, C=1, X=1	0	1

Now let us illustrate the scheme for a multistage circuit shown in Fig.5 taken from [7]. It has been shown that no two pattern test seqence exist for certain stuck-open faults in this circuit[7]. However by using the technique suggested in this paper,the circuit can be tested for all single stuck-open faults using only single test patterns.

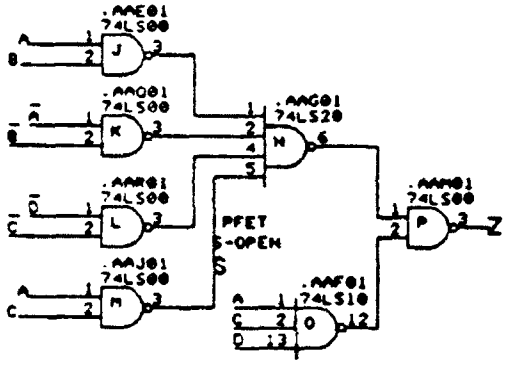

FIG.5 CMOS CIRCUIT TAKEN FROM (REF.7)

To illustrate let us derive a test for a P-transistor stuck-open fault connected to input A of the NAND gate marked 'J' in Fig5. The input pattern that will detect this fault is: A=0, B=1, C=0, D=1, X=0. When the pattern is applied, output node 'Z' will be discharged to '0' if and only if the above PFET stuck-open fault exists in the sensitized path else it will be charged to '1'. Now let us derive a test pattern for a stuck-open fault in a PFET driven by S in NAND gate marked 'N' in Fig.5 for which a three pattern test set has been suggested in [7]. This pattern is given as [ABCD]= [1001,1011,1010]. Using our approach this fault can be detected using a single test pattern :A=1, B=0, C=1, D=0, X=0. When this pattern is applied output 'Z' will be set to '0' in the absence of the stuck-open fault otherwise it will be set to '1'.

The total number of test patterns needed to detect all stuck-open faults can be derived using the following theorem.

THEOREM2: The maximum number of test patterns needed to test all stuck-open faults is equal to total number of independent paths from Vdd(GND) to output node.

Proof: If two or more transistors are in series in a path between Vdd or GND to output node, then the effect of one being stuck-open will be same as the effect of more than one stuck-open transistors connected in series. In other words, a single test pattern is sufficient to detect the presence of one or more than one stuck-open transistor in a path. Therefore, the total number of test patterns to detect all stuck-open faults cannot exceed the total number of independent paths from Vdd(GND) to the output. QED

To illustrate, let us consider circuit shown in Fig.4(b). It has four independent paths from Vdd(GND) to output node 'F'. As can be seen T2 or T3 stuck-open can be detected by just one test pattern. Similarly (T4&T6) and (T5&T6) can be tested using one test pattern for each path.

The effect of the additional transistors being faulty will be as follows:

N1 stuck-open: If N1 is stuck-open it will be easily detectable because the output of the circuit will get stuck at 0 (1) for X=0 (1). In other words the output will get stuck at the value assumed by X at that instant of time.

P1 stuck-open : If P1 is stuck open , it will not have any effect on normal operation of the circuit because P1 transfers external input to output only when CK is low during test mode. When CK is high the output is connected to Vdd or GND according to the input pattern applied to the circuit. However, if P1 gets s-open it can be detected by setting the output at 1(0) and applying X=0(1), while keeping the clock low.

P2 (N2) stuck-open : If P2(N2) is stuck open, it can be easily detected by setting the CK permanantly low, and then applying 0(1) at the additional input 'X'. The output 'F' will get stuck-at '0'(1) if P2(N2) is stuck-open.

3. Conclusion

A technique for detection of stuck-open faults in CMOS circuits has been presented. This approach needs only single test pattern for detecting such

faults, and the circuit retains its combinational characteristic. The problem of test invalidation by circuit delays, timing skews etc. has been eliminated. Only two transistors and an additional input is needed to make any gate fully testable for all single stuck-open faults. Hence the area overhead is not significant. With a slight modification and little more increase in overhead the technique can be extended to detection of s-closed faults as well.

Acknowledgement

This work was supported by Air Force of Scientific Research under grant F-49620-89-C-0069.

References

1. Y.M. El-Ziq et al.,"Functional-level test generation for stuck faults in CMOS VLSI",Proc. Int'l test conf., pp 536-546, 1981.

2. R.Chandramouli,"On testing s-open faults,"Proc. Int'l Symp. fault-tolerant computing, pp 258-265, 1983.

3. J.P. Hayes,"Fault Modelling", IEEE Design and Test of computers, pp. 88-95, April 85.

4. R.Rajsuman, A.P. Jayasuman and Y.K. Malaya,"CMOS s-open fault testability",IEEE J. Solid-state circuits, vol.-24, no.1, pp 193-194, Feb. 89.

5. R.Rajsuman, A.P. Jayasuman and Y.K. Malaya,"CMOS open fault detection in the presence of glitches and timing skews",IEEE J. Solid-state circuits, vol.-24, no..4, Aug.89.

6. R.Rajsuman, A.P.Jayasuman and Y.K. Malaya,"CMOS open fault detection using single test patterns",IEEE J. Solid-state circuits, vol.26, no.1, Jan.1991.

7. S.M. Reddy,M.K. Reddy,"Testable realization for FET s-open faults in CMOS combinational logic circuits", IEEE Trans. computers, vol. c-35, pp 742-754, Aug. 86.

8. N.K. Jha and J.A. Abraham,"Design of testable CMOS logic circuits under arbitrary delays", Vol CAD.-4, no.3, pp 264-469, July1985.

9. S.M. Reddy,M.K. Reddy and J.G. Kuhl," On testable design for CMOS logic circuits",Proc. 1983 Int. test conf., Philadelphia, PA, pp 435-445, Oct. 1983.

10. Sandip Kundu,"Design of multioutput CMOS combinational logic circuits for robust testability", IEEE Trans. on Computer-Aided-Design,Vol. 8. no 11, pp. 1222-1225, Nov.89.

Appendix C

INT. J. ELECTRONICS, 1992, VOL. 72, NO. 3, 467-470

Design of a fault-tolerant universal cell

P. K. LALA†, F. BU-SABA†, A. XIE†, and K. C. YARLAGADDA†

The design of a cell constructed from PMOS and NMOS transistors is presented. It has been designed so that it will function as a two input AND, OR and INVERTER, even in the presence of stuck-open or stuck-closed faults.

1. Introduction

It is now generally accepted that not all faults in VLSI logic can be represented by the stuck-at-0 and stuck-at-1 fault models used at the gate level. To ensure realistic modelling, faults should be considered at the transistor level, since only at this level is the complete circuit structure known. In other words, tests for circuits should be derived on the basis of possible 'shorts' and 'opens' at the transistor level (Wadsak 1978, Galiay *et al.* 1980).

A faulty transistor in a circuit can be modelled as stuck-open or stuck-closed. A stuck-open or stuck-closed transistor can be modelled by replacing the faulty transistor with an open connection or a direct short, respectively between the transistor's source and drain.

The major problem with a stuck-open fault is that it forces a combinational circuit to behave as a sequential one. The current strategy for detecting a stuck-open fault is to apply an initializing input pattern, followed by a test pattern that establishes one or more conducting paths from V_{dd} or ground to the output (Ziq *et al.* 1981, Chandramouli 1983). However, a two-pattern test can be invalidated by timing skews and charge distribution (Reddy *et al.* 1984, 1986).

2. Implementation of the universal cell

We propose the design of a universal cell constructed from NMOS and PMOS transistors, which is tolerant of stuck-closed or stuck-open faults. The structure of the universal cell is shown in Fig. 1.

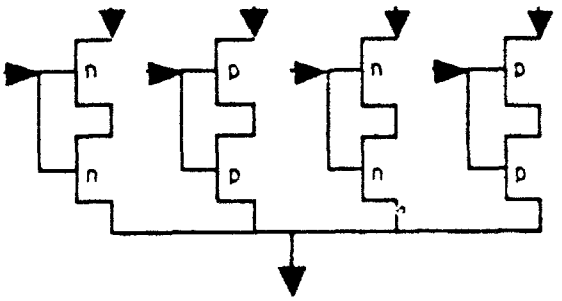

Figure 1. Universal cell structure.

Received 14 August 1991; revised 9 September 1991, accepted 10 September 1991.

†Department of Electrical Engineering, North Carolina A & T State University, Greensboro NC 27411, U.S.A.

It is constructed so that for each input pattern there are two independent signal paths from the input to the output. The cell can function as one of the conventional two input gates e.g. AND, OR, INVERTER as shown in Fig. 2.

The fault-tolerance aspect of the cell has been proved in the following lemmas.

Lemma 1

The universal cell is tolerant of any single stuck-open or stuck-closed fault.

Proof

Every pass transistor in the cell has one transistor in series and another in parallel. Thus, if a pass transistor is stuck-closed, it will not affect the correct operation of the cell because of the presence of a transistor in series with it. Similarly, if a pass-transistor is stuck-open, its effect will be masked due to the presence of another transistor in parallel with it. Hence any single transistor fault (stuck-open or stuck-closed) will have no effect on the cell. □

Lemma 2

The universal cell is tolerant of multiple stuck-on faults provided there is no more than a single fault in the same path.

Proof

We assumed that each path can only have a single fault; therefore, multiple stuck-on faults are equivalent to single stuck-on faults on different paths. Since all paths are independent, and by Lemma 1 the universal cell is tolerant of single stuck-

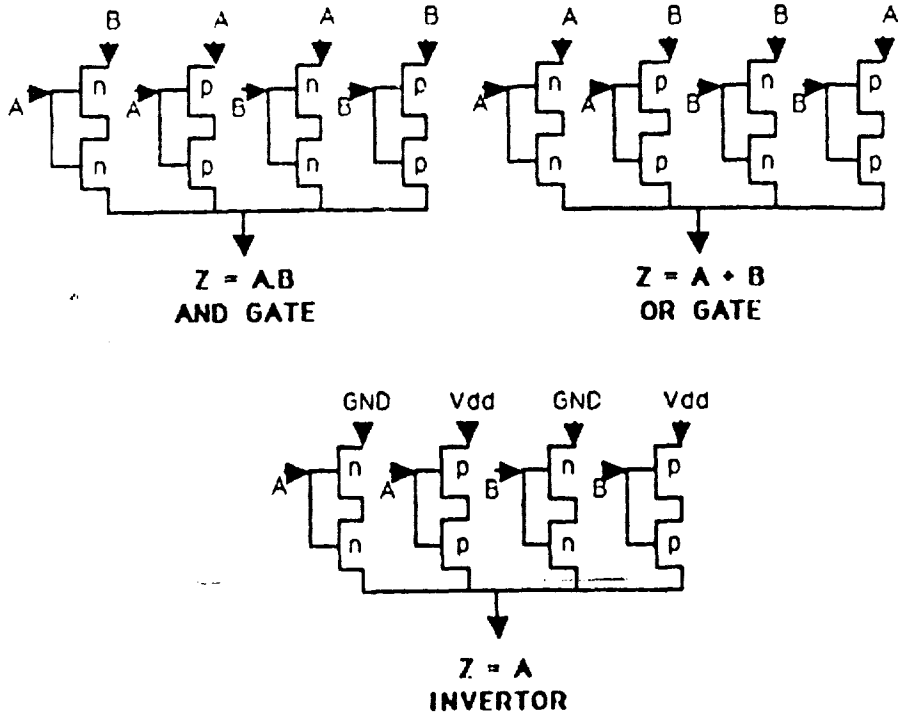

Figure 2. Fault-tolerant two-input gates

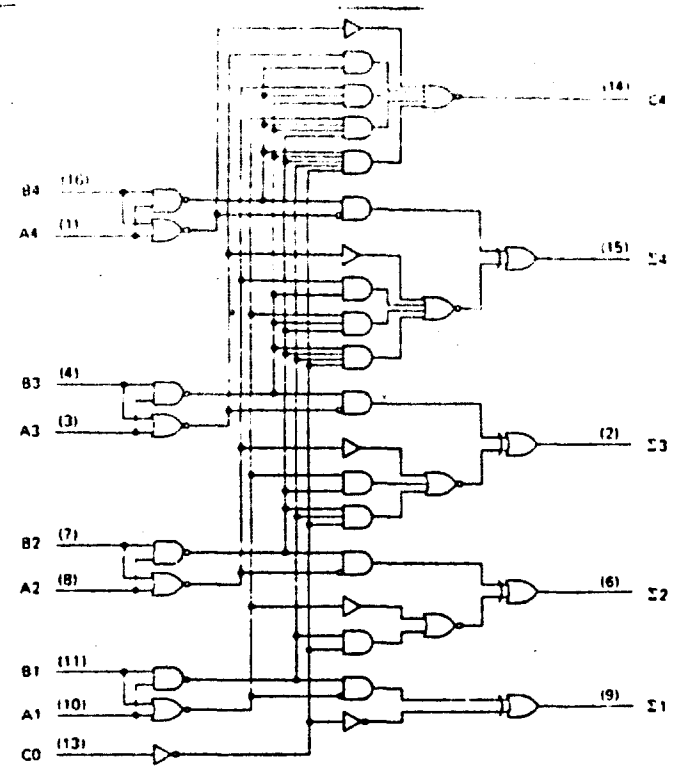

Figure 3. 4-bit binary full adder with fast carry (courtesy of Texas Instruments).

on faults, the multiple stuck-on faults will have no effect on the correct operation of the cell. □

Lemma 3

The universal cell is tolerant of multiple stuck-open faults provided there are no faults on one of the two signal paths driving the output.

Proof

There are two signal paths to the output for each input combination. If either or both the transistors in one signal path are stuck-open, the path is disconnected from the output. Thus both transistors in the other signal path have to remain fault free in order for the cell to remain operational. □

3. Conclusions

Logic circuits constructed by using the proposed universal cell will be tolerant of both stuck-closed and stuck-open faults. A 4-bit binary adder with fast carry, Texas Instruments SN74283 shown as in Fig. 3, has been implemented using this fault tolerant universal cell. The circuit has been laid out using the MAGIC layout editor and simulated using the various simulation tools like e-sim(logic simulator) and CaZM (timing simulator). The layout size of the circuit is $355\lambda \times 364\lambda$ compared to

that designed using the standard cell approach ($314\lambda \times 364\lambda$). Currently, as mentioned above, stuck-open faults are detected off-line by applying a two-pattern test sequence. In the proposed universal cell, single stuck-open faults will be masked on-line thus eliminating the need for testing such faults. Single stuck-closed faults in a cell will also have no effect on a circuit during its normal operation.

Acknowledgment

This work was supported by the US Air Force Office Of Scientific Research under Grant F49620-89-C-0069.

References

Chandramouli, R., 1983, On testing stuck-open faults. *Proceedings International Symposium on Fault-Tolerant Computing*, Milan, pp. 258-265.

Galiay, J., Crouzet, Y., and Vergniault, M., 1980, Physical versus logical fault models in MOS LSI circuits, impact on their testability. *I.E.E.E. Transactions on Computers*, **29**, 527-531.

Reddy, S. M., and Reddy, M. K., 1986, Testable realizations for FET stuck-open faults in CMOS combinational logic circuits. *I.E.E.E. Transactions on Computers*, **35**, 742-754.

Reddy, S. M., Reddy, M. K., and Agarwal, V. D., 1984, Robust tests for stuck-open faults in CMOS combinational logic circuits. *Proceedings of the International Symposium on Fault-Tolerant Computing*, Kissemmee, Florida, pp. 44-49.

Wadsak, R. L., 1978, Fault Modeling and Logic Simulation of CMOS and NMOS integrated circuits. *Bell System Technical Journal*, **57**, 1449-1474.

Ziq, Y. M. El, and Cloutier, R. J., 1981, Functional-level Test generation for stuck-open faults in CMOS VLSI. *Proceedings International Test Conference*, Philadelphia, pp. 536-546.

Appendix D

A NEW TECHNIQUE FOR TOTALLY SELF-CHECKING CMOS CIRCUIT DESIGN FOR STUCK-ON AND STUCK-OFF FAULTS.

Manjit S. Cheema, Member, IEEE, AND P.K. Lala, Senior Member, IEEE

Department of Electrical Engineering,
North Carolina A&T State University, Greensboro, N.C. 27411.

Abstract

This paper presents a new technique for designing single stage fully complementary metal oxide semiconductor (FCMOS) and CMOS domino logic circuits so that they are totally self checking for all single s-off and s-on faults. It involves the encoding of the output of the circuit in an error detecting code. CMOS circuits designed using the technique have two outputs. Two of the combinations (01, 10) are considered to be valid code-words.. The circuit is augmented such that any stuck-off (stuck-on) fault in the modified circuit produces a non-valid output 11 (00), thus ensuring automatic fault detection.

Index terms: Self-checking, fault secure, self testing, code-word, stuck-off and stuck-on faults.

Introduction

As digital systems become more complex, the necessity to have systems that have the capability of self checking is growing. Self checking can be defined as the ability to verify automatically, whether there is any fault in the logic (chips, boards or assembled systems) without the need for externally applied test stimuli. Totally self checking circuits are very desirable for highly reliable digital system design, since all faults from a given set would cause a detectable, erroneous output. A technique is presented in this paper for modifying CMOS circuits so that they will be totally self checking for all single transistor stuck-off (s-off) and stuck-on (s-on) faults. Significant amount of research have been carried out in the area of testable CMOS designs [1,2,3,4,5], but not much have been reported on self-checking CMOS design [6,7]. CMOS has emerged as the dominant technology for manufacturing digital systems. A CMOS circuit consists of a P-network connected between Vdd and the output node, and a N-network connected between GND and the output node (Fig.-1(a)). The circuit for P-network and N-network bear a dual relationship by DeMorgan's theorem. This type of CMOS circuits is known as FCMOS. An alternative method of designing CMOS circuits is to use the precharged principle. Such circuits are called CMOS domino logic circuits.

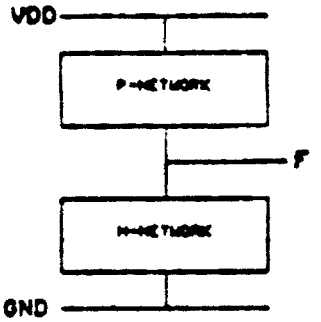

FIG.1(A). BLOCK DIAGRAM OF FCMOS CIRCUIT

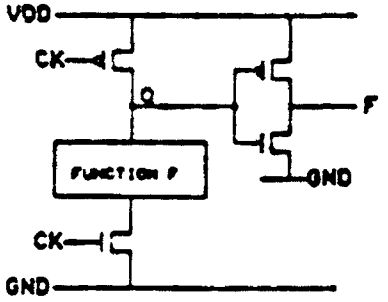

FIG.1(B) BLOCK DIAGRAM OF DOMINO CMOS CIRCUIT

A CMOS domino logic circuit consists of a n-network for implementing the function and a clocked p-channel and a clocked n-channel transistor (Fig.1(b)). Also a CMOS inverter is connected at the output to make it low during the precharge phase. The output node Q is precharged to 1 when the clock is low. During the evaluate phase i.e. when clock is high, if the input pattern closes the path between GND and output node Q, the output is pulled to 0 otherwise it stays at 1.

It has now been generally accepted that not all faults in CMOS VLSI logic can be represented by the stuck-at-0 and stuck-at-1 models used at the gate

0-7803-0623-6/92 $3.00

level. In order to ensure realistic modeling, faults should be considered at the transistor level, since only at this level the complete circuit structure is known. In other words, circuits should be tested for 'shorts' and 'opens' at the transistor level[8,9]. We consider two types of faults in this paper: stuck-off (s-off) and stuck-on (s-on). A s-off transistor fault implies the permanent opening of the connection between source and drain of the transistor. On the other hand, a s-on transistor fault implies the permanent closing of the path between the source and the drain of the transistor. It should be noted that a s-off or s-on transistor fault does not mean that the input line connected to the transistor in question is stuck-at 1 or 0, it is only the transistor itself that is considered to be s-off or s-on. A s-off transistor fault causes the output to be connected neither to Vdd nor to GND. On the other hand, a s-on fault causes the output to be connected to both Vdd and GND and hence result in a short circuit condition. It should be noted that the terms stuck-off and stuck-open (stuck-on and stuck-closed) are not interchangeable. In a stuck-open transistor, the drain source resistance is significantly higher than the off-resistance of a non-faulty transistor, whereas the drain source resistance of a stuck-off transistor is approximately equal to the off-resistance of non-faulty transistor. A stuck-on transistor has the same drain source resistance as the on-resistance of a fault-free transistor,whereas a stuck-closed transistor exhibits the drain-source resistance which is significantly lower than the normal on-resistance. A stuck-off or stuck-on transistor fault can be modeled by replacing the faulty transistor with an open connection or a direct short respectively,between the transistor's source and drain.

The following two definitions describe the manner in which self checking circuits deal with faults [10].

Fault secure: A circuit is fault secure for a given set of faults, if for any fault in the set the circuit never produces an incorrect code word at the output for the input code space.

Self testing: A circuit is self testing, if for every fault from a given set of faults, the circuit produces a non-code word at the output for at least one input code word.

A circuit is said to be **totally self checking** if it satisfies both the above properties i.e. it is both fault secure and self testing.

Self-checking implementation of CMOS domino logic circuits

In this section we present a new technique for making any single stage CMOS domino logic circuit totally self checking for all single stuck-off(s-off) and stuck-on (s-on) faults. The block diagram of the proposed self checking design for domino logic circuit is shown in Fig.2 . The original domino logic circuit

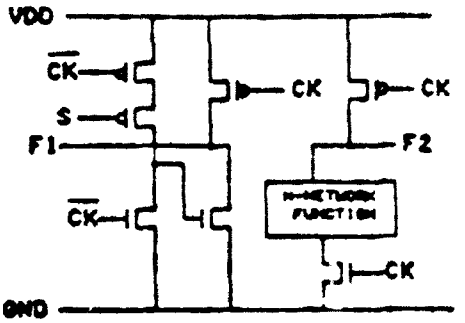

FIG.2 BLOCK DIAGRAM OF TOTALLY SELF CHECKING DESIGN OF DOMINO CMOS CIRCUIT

has been augmented using three extra transistors, and an external input S which will be set to 1(0) if output F1F2=10 (01) has to be produced during normal operation. The output of the circuit is encoded in 1-out-of-2 code. A non-codeword (00 or 11) at the output of the circuit indicates the presence of a fault in the circuit. For any fault (s-off or s-on) in the circuit, both output lines F1 and F2 will assume a value of '0' or both assume a value of '1'. In Fig.2 the circuit output F1/F2 are charged to 1 during the precharge phase i.e. when the clock is low. Once the clock CK goes high, outputs (F1/F2) will be code-words i.e. either '01' or '10' if there is no fault in the circuit. For any transistor s-off(s-on) fault in the circuit the outputs will remain at the non-code word 11, and for a s-on fault they will be discharged to the non-codeword 00.

Lemma-1: A CMOS domino logic circuit modified as shown in Fig. 2 is fault secure for all single s-off and s-on faults.

Proof: When an input pattern is applied to domino logic circuit, the circuit outputs F1/F2 are charged to 11 during the precharge phase i.e. when the clock is low . During the evaluate phase i.e. when clock CK goes high, either Vdd or Gnd is connected to the output node and set the output to 01 or 10 value. The outputs will only change to '01' or '10' iff there is no fault (s-off or s-on) in the conducting path activated by an input pattern. A path between Vdd (GND) and output node is said to be activated by an input pattern if all the transistors present in that path are turned on. If a path connecting Vdd (GND) to output node is activated and a s-off fault is present in the activated path , the output will remain at a non-code-word value 11. On the other hand if an input

pattern activates a path between Vdd (GND) and output node and a s-on fault is present in n-part (p-part) that enables a path between GND (Vdd) and output node, the circuit outputs will assume a value 00. Since 11 and 00 are non-code-words, the presence of the fault will be detected. Therefore, for any single fault in the circuit the output is always set to a non-code-word value, and never to an incorrect code-word i.e. a 01 is not changed to 10 or vice versa. Hence the circuit is fault secure. QED.

Lemma-2: A CMOS domino logic circuit modified as shown in Fig. 2 is self testing for all single s-off and s-on faults.

Proof: In domino logic there is just one conducting path from Vdd to output node. Therefore, the input pattern that connects Vdd to output node will be a test for a fault in that path. In other words in the presence of a fault the outputs will either remain unchanged at 11 or change to 00. For a circuit with two or more conducting paths from GND to output node, it is possible that an input pattern may activate more than one conducting path from GND to output node. Therefore if a fault is present in one path, it might not be detected with such an input pattern. However, this does not mean that the fault is undetectable, because for each conducting path there exists a unique input pattern that enables only that path and disables all other paths from GND to output node. Such an input pattern will definitely set the outputs to a non-code word value 11 or 00 in the presence of a fault (s-off or s-on) in the conducting path.. In other words there exists a test for each possible single s-off or s-on fault in the circuit; hence the circuit is self testing. QED.

Lemma-3: Any CMOS domino logic circuit augmented as shown in Fig. 2 is totally self checking for all s-off and s-on faults.

Proof: For any circuit to be totally self checking , it has to be both self testing and fault secure. As proved above in lemma-1 and lemma-2, the augmented versions any CMOS domino logic circuit is both self testing and fault secure. Hence the augmented design is totally self checking. QED.

To illustrate the proposed technique let us consider the domino CMOS circuit shown in Fig. 3 that implements the function $F=AB+C(A+B)$. The augmented circuit is shown in Fig. 4. Let us suppose transistor T1 or T2 is s-off. When the input pattern S=0 A=0 B=0 C=0 is applied to the circuit ,outputs F1/F2 will go to '01' if and only if transistor T1 or T2 is not s-off otherwise they will remain at '11'. Next let us consider a s-on fault. e.g. T5 s-on. When the input pattern : A=1 B=0 C=0 S=0 is applied , circuit outputs F1/F2 will assume the value '00' if T5 is s-on otherwise will be set to '01'. Input patterns

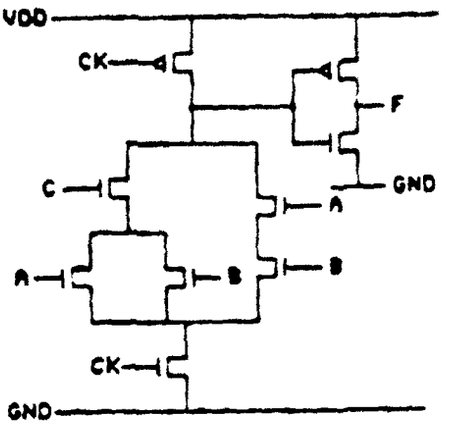

FIG.3. CMOS DOMINO LOGIC CIRCUIT FOR FUNCTION F=AB+C(A+B)

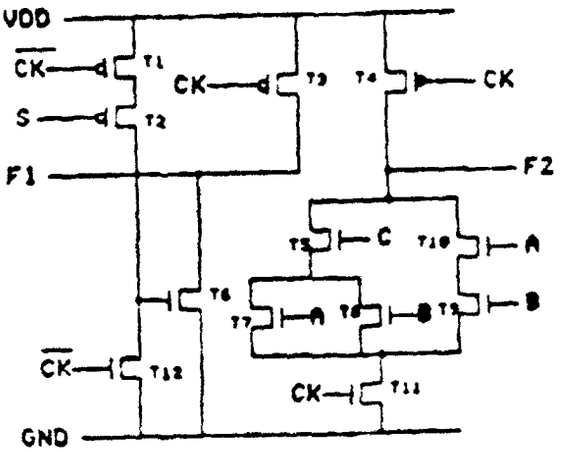

FIG.4 TOTALLY SELF CHECKING DOMINO CMOS CIRCUIT FOR FUNCTION F=AB+C(A+B)

which detect certain single s-off and s-on fault during normal operation in the circuit of Fig. 4 are given below:

Fault	Input Pattern	Faulty		Fault free	
		F1	F2	F1	F2
T9 s-off	A=1,B=1,C=0,S=1	1	1	1	0
T7 s-off	A=1,B=0,C=1,S=1	1	1	1	0
T5 s-on	A=1,B=0,C=0,S=0	0	0	0	1
T10 s-on	A=0,B=1,C=0,S=0	0	0	0	1

Self checking design of FCMOS circuits

This section of the paper deals with the totally self checking design of FCMOS circuits. The block diagram of the proposed self checking design is shown in Fig. 5. The modification suggested in Fig. 5 for FCMOS circuit involves the addition of just six extra

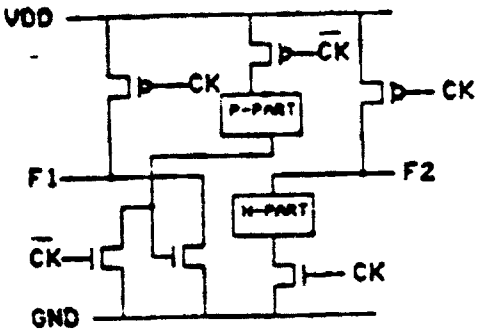

FIG.5. TOTALLY SELF CHECKING DESIGN OF CMOS CIRCUIT

transistors, no additional input is required. As stated in the previous section, a non-codeword (00 or 11) at the output of the circuit indicates the presence of a fault in the circuit.

Lemma-4: Any FCMOS circuit augmented as shown in the Fig. 5 is fault secure for all s-off and s-on faults in the circuit

Proof: In the proposed design, when the circuit receives an input pattern and the clock is low, both outputs F1 and F2 are charged to 1. When the clock goes high, either Vdd or GND is connected to one of the output nodes. This results in discharging of the respective output node, thus producing 01 or 10 output. For any s-off transistor in the path activated by an input pattern, the circuit will always produce F1=F2=1, a non-code output thus indicating the presence of a fault. Suppose an input pattern activates a path in p-part (n-part) and a s-on transistor is present in n-part (p-part). If this s-on transistor results in activating a path between GND (Vdd) and output node, the circuit will produce F1=F2=0, a non-code output indicating the presence of the s-on transistor. Since for any fault, the circuit never produces an incorrect code word i.e. 10 instead of 01 or vice versa, the circuit is fault secure for all s-off and s-on faults. QED.

Lemma-5: Any FCMOS circuit augmented as shown in the Fig. 5 is self testing for all s-off and s-on faults in the circuit.

Proof: In an FCMOS circuit, it is possible for an input pattern to activate more than one path from Vdd (GND) to output node. Therefore, if a fault is present in one of the paths it may remain undetected for such an input pattern. However, it can be easily verified that each distinct path from Vdd (GND) to output node has at least one unique input pattern that activates only that path and disables all other paths. Therefore, when such an input pattern is received, the s-off (s-on) fault present in the activated path will manifest by producing a non-code value 11 (00) at the output. Since for every fault there is at least one input pattern for which the the resulting output is a non-code word, the circuit is self testing. QED.

Lemma-6: Any FCMOS circuit augmented as shown in Fig. 5 is totally self checking for all s-off and s-on faults.

Proof: A circuit is said to be totally self checking if it is both fault secure and self testing. The modified circuit as proved above in lemma-4 and lemma-5 is both self testing and fault secure for all s-off and s-on transistor faults. Therefore the augmented design is totally self checking for all s-off and s-on faults. QED

As an example of the proposed technique let us consider the CMOS circuit shown in Fig. 6 that implements the function $F = \overline{(A+B)(B+C)(C+A)}$. The self checking design of the circuit is shown in Fig. 7.

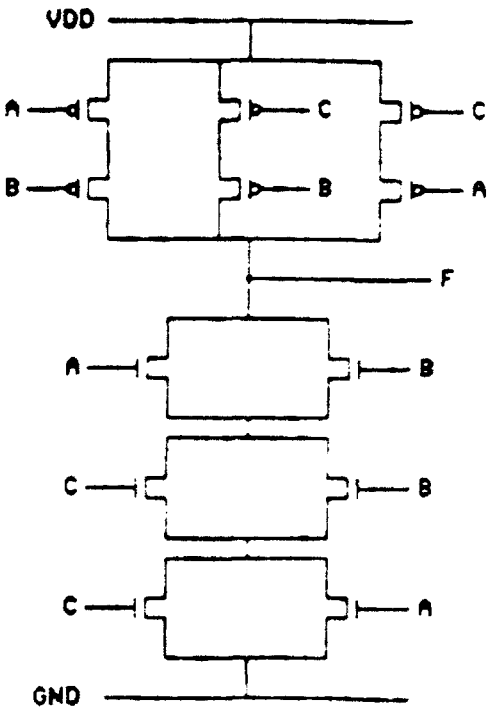

FIG.6. CMOS CIRCUIT FOR THE FUNCTION $F=\overline{(A+B)(B+C)(C+A)}$

In this circuit when the input pattern (A=0 B=1 [illegible]) is received, all three conducting paths from Vdd [illegible] output node are closed. Therefore if a fault occurs in

any of the paths, it will not be detected with this input pattern. However, input pattern A=0 B=0 C=1, will acuvate the path with transistor T2 and T3 only.

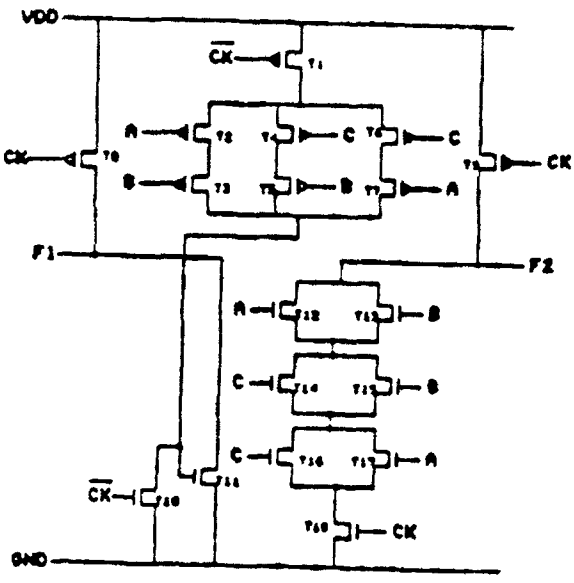

FIG. 7 TOTALLY SELF CHECKING CMOS CIRCUIT REALIZING FUNCTION F=AB+BC+CA

Therefore any fault in this path will be detected by this input pattern. Similarly A=1 B=0 C=0 will acuvate the path with transistors T4 & T5 and hence will detect any fault in this path. Let us suppose transistor T4 is s-off. When an input pattern such as:A=1,B=0,C=0 is received, outputs will be F1=0,F2=1 if and only if T4 is not s-off otherwise they will remain at non-codeword 11. Let us consider a s-on fault at T16. When an input pattern such as :A=0,B=1,C=0 is received, outputs F1/F2 will be discharged to 00 iff T16 is s-on otherwise they will be set to '01'. Input pattern for certain s-off and s-on faults in the circuit shown in Fig. 7 are given below:

Fault	Input Pattern	Faulty		Fault free	
		F1	F2	F1	F2
T2 s-off	A=0,B=0,C=1	1	1	0	1
T 15 s-off	A=1,B=1,C=0	1	1	0	1
T5 s-on	A=1,B=1,C=0	0	0	1	0
T14 s-on	A=1,B=0,C=0	0	0	0	1

It should be mentioned that in order to avoid a direct short between Vdd and GND if any of the three transistors T8,T9 or T10 is s-on, a transistor (p-transistor in p-part and a n-transistor in n-part) may be connected in series with each of the following transistors: T8,T9 and T10. Since only single s-on faults have been assumed, the possibility of a short circuit due to a single s-on fault does not exist.

Conclusion

We have proposed a technique for designing CMOS circuits so that they are totally self checking for all single s-off and s-on faults. This has been achieved by adding just a few transistors without affecting the speed of the circuit. The problem of circuit delays and test invalidation have been eliminated. In the circuits designed using the proposed technique all single s-off or s-on faults are detected automatically without applying any external test stimuli. Application of the technique to mulustage CMOS circuits is under investigation and the result will be presented in a future paper.

Acknowledgement: This work was supported by Air Force of Scientific research under grant F 49620-89-C-0069.

References:

1. Y.K. Malaya,"Testing stuck-on faults in CMOS integrated circuits", in Proc. Int. Conf. Computer-Aided-Design (Santa Clara), CA), Nov. 1984, pp. 248-250.

2. Vojin G.Oklobdzija and Predrag G. Kovijanic,"On testability of CMOS domino logic", in Proc. Int. Symp. Fault-tolerant Compuung (Orlando, FL), June 1984, pp. 50-55..

3. S.M. Reddy and M.K. Reddy,"Testable realizauons for FET stuck-open faults in CMOS combinauonal logic circuits", IEEE Trans. on Computers, Vol. C-35, No. 8, August 1986.

4. N.K. Jha,"Testing for muluple faults in domino CMOS logic circuits", IEEE Trans. Computer-Aided-Design, Vol. 7, pp. 109-116, Jan.1988.

5. S.M. Reddy and M.K. Reddy, and J.G. Kuhl,"On testable design for CMOS logic circuits", in Proc. Int. Test Conf. (Philadelphia,PA), Oct. 1983,pp. 435-445.

6. S.R. Manthani and S.M. Reddy,"On CMOS totally self-checking circuits", in Proc. Int. Test Conf. (Philadelphia, PA), Oct. 1984, pp-866-877.

7. N.K.Jha and J.A. Abraham,"Totally self checking MOS circuits under realistic physical failures", in Proc. Int. Conf. Computer Design (Port Chester,NY), Oct. 1984, PP. 665-670.

8. J.Galiay et al.,"physical versus Logical fault models in LSI circuits:impacts on their testability",IEEE Trans. Comp. Vol.C-29,No.6,June80,pp.527-531.

9. Y.M. El-Ziq et al.,"Functional-level test generaton for stuck faults in CMOS VLSI",Proc. Int'l Test Conf.,1981,pp 536-546.

10. Anderson,D.A and G. Metze,"Design of totally self checking check circuits for m-out-of-n codes",IEEE Trans. on Computers,March 73,pp-263-269.

Totally Self-Checking CMOS Circuit Design for Breaks and Stuck-on Faults

Manjit S. Cheema and P. K. Lala

Abstract—**This paper presents a new technique for designing totally self-checking FCMOS circuits. Two types of defects have been considered, e.g., breaks (caused by missing conducting material or extra insulating material) and transistor stuck-on faults. In order to make FCMOS circuits totally self-checking for all breaks and transistor stuck-on faults, only four extra transistors need to be added to the functional circuit. The additional circuitry is added in such a way that for any break or transistor stuck-on defect in the functional circuit, the outputs assume a value of 01 or 10, respectively. The output of the defect-free circuit will be 11 (00) when the input pattern applied to the circuit connects V_{dd} (GND) to the output node.**

I. Introduction

IN recent years the complexity of digital systems has increased dramatically. Due to the increase in number of devices on a chip, the controllability and observability of a system are decreasing. Also, it is almost impossible not to have faults somewhere in a system at any given time. Totally self-checking circuits are very desirable for highly reliable digital system design, since during normal operation all faults from a given set would cause a detectable erroneous output.

Currently CMOS is the dominant technology for very-large-scale integration of digital systems. The realistic modeling of the defects in CMOS VLSI logic can be done only at the transistor level, since only at this level is the complete circuit structure known [1]–[4]. It has been established that breaks and transistor stuck-on (s-on) faults constitute a significant portion of the defects occurring in CMOS circuits [5]. A s-on transistor fault implies the permanent closing of the path between the source and the drain of the transistor. It should be noted that a s-on transistor fault does not mean that the input line connected to the transistor in question is stuck-at 1 or 0, it is the transistor itself that is considered to be s-on. A s-on fault in the n-network can be modeled by setting the input to the transistor under test permanently at 1, and a s-on fault in the p-network can be modeled by setting the input to the faulty transistor permanently at 0.

Breaks can be caused by either missing conducting material or extra insulating material [6]. Break defects in CMOS circuits can be of two kinds, e.g., intragate breaks and signal line breaks. Fig. 1 shows the possible breaks

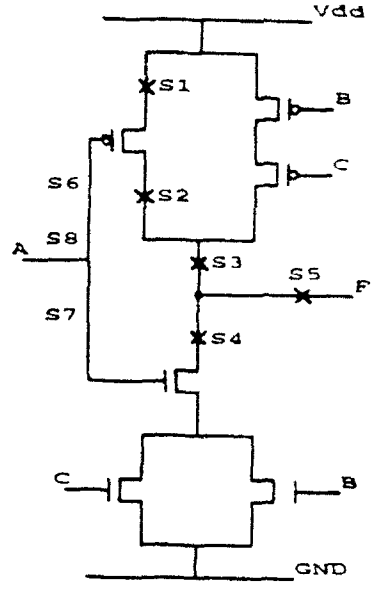

Fig. 1. FCMOS circuit showing break defects.

in an FCMOS circuit. Intragate breaks occur internal to a gate, e.g., break in source line ($S1$), break in drain line ($S2$), break between p-network and output node ($S3$), break between n-network and output node ($S4$), or break disconnecting both the p network and n-network from the output node ($S5$). Signal line breaks in FCMOS can either make the gate of only a p-transistor ($S6$) or of an n-transistor ($S7$) float. It is also possible that gates of both transistors may float ($S8$) in which case one transistor may conduct and the other remains in a nonconducting state [7]. The capacitive coupling between adjacent nodes may further complicate the situation [8]. It has been observed that if a transistor is in a nonconducting state due to a signal line break ($S6$, $S7$, or $S8$), the circuit will behave as if it has an intragate break. A break in a signal line ($S6$, $S7$, or $S8$) can also cause a transistor to be stuck-on [6]. It has been shown in [7] that a transistor with a break in the source line can still conduct and hence pass the desired signal. Therefore, such defects can be modeled as transistor stuck-on faults.

Most of the published literature on CMOS testing deal with testable design [9]–[11]; not much has been reported on the self-checking design of CMOS circuits [12], [13]. In this paper a new technique is presented for modifying single-stage and multistage FCMOS circuits so that they will be totally self-checking for all single intragate breaks ($S1$–$S5$) and transistor s-on faults. In the following sections any reference to breaks will mean intrabreaks only.

Manuscript received November 26, 1991; revised February 26, 1992. This work was supported by the Air Force Office of Scientific Research under Grant F-49620-89-C-0069.

The authors are with the Department of Electrical Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411.

IEEE Log Number 9200638.

The following two definitions describe the manner in which self-checking circuits deal with faults [14].

Fault Secure: A circuit is fault secure for a given set of faults if for any fault in the set the circuit never produces an incorrect code word at the output for the input code space.

Self-Testing: A circuit is self-testing if for every fault from a given set of faults the circuit produces a noncode word at the output for at least one input code word.

A circuit is said to be *totally self-checking* if it satisfies both the above properties, i.e., it is both fault secure and self-testing.

II. Totally Self-Checking Design of FCMOS Circuits

In this section we present a new design technique for designing FCMOS circuits so that they will be totally self-checking for all breaks and transistor s-on faults. Fig. 2 shows the block diagram of the proposed totally self-checking design. As can be seen from the diagram the conventional design has been augmented using just four extra transistors. In the proposed design configuration, $F1$ ($F2$) is labeled as the output of p-part (n-part). When the clock is low $F1$ is discharged to 0 and $F2$ is charged to 1. The extra transistors are connected in such a way that for any defect (break or s-on) in the circuit the outputs will assume a value of 01 or 10. For a fault-free circuit the outputs will assume a value 11 (00) if the input pattern activates a path in the p-network (n-network).

Theorem 1: Any FCMOS circuit augmented as shown in the Fig. 2 is totally self-checking for all single breaks and s-on faults.

Proof: When an input pattern is received by the circuit with the clock CK low, outputs $F1/F2$ are set to 01. When the clock goes high, either V_{dd} or GND is connected to one of the output nodes and the outputs $F1/F2$ are changed to 11 or 00 if there is no fault in the circuit. A path between V_{dd} (GND) and output node is said to be activated by an input pattern iff all the transistors present in that path are turned on. For a break in a conducting path, the outputs $F1/F2$ will remain at 01 when an input pattern is applied that activates this path. Alternatively, if an input pattern activates a conducting path in the p-part and a s-on fault in the n-part falsely activates a path, the output shall assume a value 10. No single defect (break or s-on) in the circuit can change the output from 00 to 11 or vice versa. Hence, the circuit is fault secure for all breaks and s-on faults. Furthermore, in FCMOS it is possible that an input pattern activates more than one path from V_{dd} (GND) to output node. In such a situation a defect may be present in one of conducting paths but its effect may not appear at the output of the circuit. This does not mean that the defect is redundant or undetectable. For every possible conducting path in a FCMOS circuit, there is at least one distinct input pattern that activates only that path and disables all other paths. When such an input pattern is received by the circuit, the outputs assume a 01 or 10 value and hence the fault is detected. Therefore, it can be concluded that for every possible defect (break or s-on) in the FCMOS circuit there exists an

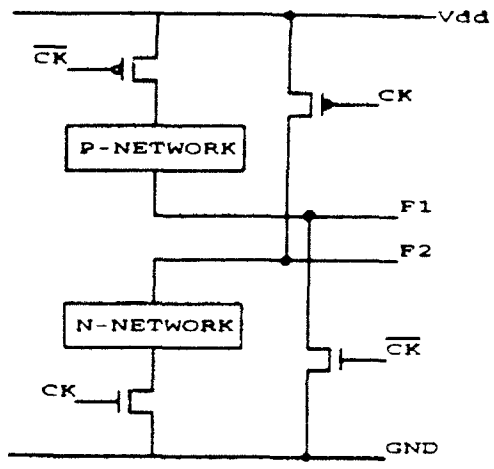

Fig. 2. Block diagram of totally self-checking FCMOS circuit.

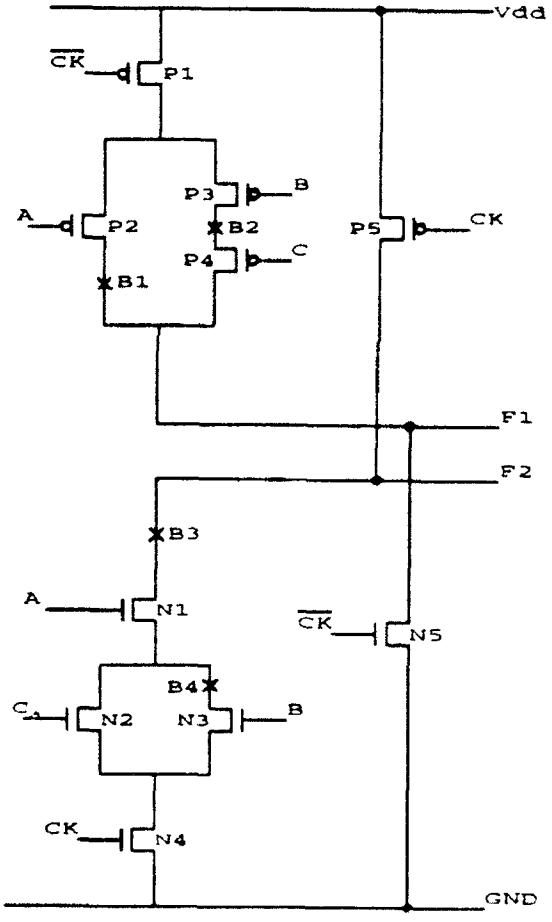

Fig. 3. Totally self-checking FCMOS circuit implementing function $\overline{A(B+C)}$.

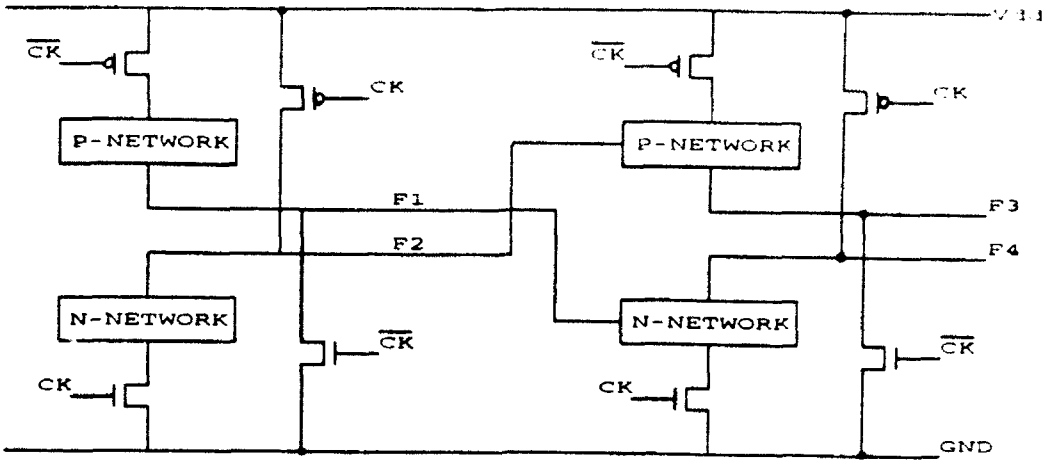

Fig. 4. Block diagram of a totally self-checking two-stage FCMOS circuit.

input pattern that will detect the fault. Hence, the circuit is self-testing for all single breaks and s-on faults. Since the circuit is both self-testing as well as fault secure, it is totally self-checking. Q.E.D.

Let us apply the technique to the CMOS circuit shown in Fig. 1. The circuit implements the function $F = [\overline{A(B + C)}]$. The modified version of the circuit is shown in Fig. 3. To illustrate the self-checking capability of the circuit of Fig. 3, let us assume a break at $B2$ in the p-part of the circuit. When an input pattern such as $A = 1$, $B = 0$, $C = 0$ is applied, outputs $F1/F2$ will be set to 01 when the clock is low. When the clock goes high both will assume the value 1 if and only if there is no break in the activated path; otherwise they will remain at 01. Next we consider a s-on transistor $N1$. When an input pattern such as $A = 0$, $B = 0$, $C = 1$ is applied, outputs $F1/F2$ will assume the value 10 iff $N1$ is s-on; otherwise both the outputs will be charged to 1. Input patterns for some possible breaks and s-on faults in the circuit of Fig. 3 are given below:

Fault	Input Pattern	Faulty output		Fault-free output	
		$F1$	$F2$	$F1$	$F2$
Break at $B1$	$A = 0, B = 1, C = 1$	0	1	1	1
Break at $B4$	$A = 1, B = 1, C = 0$	0	1	0	0
$P3$ s-on	$A = 1, B = 1, C = 0$	1	0	0	0
$N2$ or $N3$ s-on	$A = 1, B = 0, C = 0$	1	0	1	1

The self-checking implementation of a multistage FCMOS circuit is shown in Fig. 4. As can be seen, the output of p-network (n-network) of the first stage is connected to the n-network (p-network) of the second stage. Since the outputs $F1/F2$ of the first stage assume a value 01 when the clock is low, both the n-type and p-type transistors of the second stage will remain off. When the clock goes high, output $F1/F2$ will become either 11 or 00 and turn on either the n-transistor or the p-transistor. In the presence of a break, since the output of a stage remains at 01, the succeeding stage transistors will remain off and hence the fault will be propagated to the final output. Alternatively, for a s-on fault the output of a stage assumes the value 10. Therefore, both the transistors of the succeeding stage connected to the output of the preceding stage will turn on and make its output 10. Hence, the fault will be detected.

V. Conclusion

A technique for designing totally self-checking CMOS circuits has been presented in this paper. As far as the authors are aware no technique is currently available for the design of self-checking CMOS circuits which considers breaks and s-on faults. Self-checking design of static (FCMOS) circuits has been considered for detecting breaks and s-on faults. Application of the technique to any FCMOS circuit provides concurrent checking of all single intragate breaks and s-on transistor faults in the circuit. This has been achieved at the expense of very little overhead in terms of transistors. Moreover, the speed of the circuit is not affected.

References

[1] R. L. Wadsack, "Fault modelling and logic simulation of CMOS and MOS integrated circuits," *Bell Syst. Tech. J.*, vol. 57, pp. 1449-1474, May-June 1978.

[2] W. Maly, "Realistic fault modeling for VLSI testing," in *Proc. 24th ACM/IEEE Design Automation Conf.*, 1987, pp. 173-180.

[3] J. P. Hayes, "Fault modeling," *IEEE Design Test Comput.*, pp. 88-95, Apr. 1985.

[4] F. J. Ferguson and J. P. Shen, "A CMOS fault extractor for inductive fault analysis," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 1181-1194, Nov. 1988.

[5] F. J. Ferguson and J. P. Shen, "Extraction and simulation of realistic CMOS faults using inductive fault analysis," in *Proc. Int. Test Conf.*, 1988, pp. 475–484.

[6] F. J. Ferguson, M. Taylor, and T. Larrabee, "Testing for parametric faults in static CMOS circuits," in *Proc. Int. Test Conf.*, 1990, pp. 436–442.

[7] W. Maly, P. K. Nag, and P. Nigh, "Testing oriented analysis of CMOS ICs with opens," in *Proc. Int. Conf. Computer-Aided Design*, 1988, pp. 344–347.

[8] J. M. Soden, R. K. Treece, M. R. Taylor, and C. F. Hawkins, "CMOS IC stuck-open fault electrical effects and design considerations," in *Proc. Int. Test Conf.*, 1989, pp. 423–429.

[9] N. K. Jha and J. A. Abraham, "Design of testable CMOS logic circuits under arbitrary delays," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, no. 3, pp. 264–269, July 1985.

[10] S. Kundu, "Design of multioutput CMOS combinational logic circuits for robust testability," *IEEE Trans. Computer-Aided Design*, vol. 8, no. 11, pp. 1222–1225, Nov. 1989.

[11] S. M. Reddy and M. K. Reddy, "Testable realizations for FET stuck-open faults in CMOS combinational logic circuits," *IEEE Trans. Comput.*, vol. C-35, no. 8, Aug. 19[illegible].

[12] S. R. Manthani and S. M. Reddy, "On CMOS totally self-checking circuits," in *Proc. Int. Test Conf.* (Philadelphia, PA), Oct. 1984, pp. 866–877.

[13] N. K. Jha and J. A. Abraham, "Totally self checking MOS circuits under realistic physical failures," in *Proc. Int. Conf. Computer Design* (Port Chester, NY), Oct. 1984, pp. 665–670.

[14] D. A. Anderson and G. Metze, "Design of totally self checking check circuits for m-out-of-n codes," *IEEE Trans. Comput.*, vol. C-22, pp. 263–269, Mar. 1973.

Appendix E

A Concurrent Checking Scheme for Single and Multibit Errors in Logic Circuits

B.Kolla, P.K.Lala and K.C.Yarlagadda

Department of Electrical Engineering
N.C A&T State University
Greensboro, N.C 27411.

Abstract

A new scheme for detecting single and multibit (unidirectional and bidirectional) errors using residue codes has been proposed. This procedure has been applied to circuits with outputs upto 8 bits. It has been shown that about 99% of all multibit errors can be detected using this scheme.

1. Introduction

With the increase in complexity and density of VLSI chips, transient/intermittent faults are emerging as the dominant failure mode in VLSI circuits[1]. Existing off-line test strategies are incapable of detecting these types of faults since they have been designed to detect permanent faults. Thus there is growing need to develop new test strategies to detect transient/intermittent faults, which require continuous monitoring of circuits. This is known as concurrent or on-line checking; circuits with concurrent checking capability are known as self-checking circuits.

Concurrent checking circuits are typically designed using coding techniques[2].The output bits of a circuit are encoded such that in the presence of a fault the circuit produces a non-valid code word. Typically used coding techniques are parity code, Berger code, m-out-of-n code and residue codes.

It has been shown that mod 3 residue code can be incorporated cost-effectively into VLSI designs for the detection of errors resulting from transient and permanent faults[3]. Rules based on mod3 residue code have been presented to design circuits for on-line error detection[4]. In this paper we present a new scheme for concurrent error detection using mod 7 residue code. In general low cost mod values e.g 7,15, 31, 63 are used to calculate residues. The concurrent checking scheme proposed in this paper does not alter the original design and can be used to make any arbitrary circuit self-checking. Fig 1 illustrates the proposed scheme for concurrent checking.

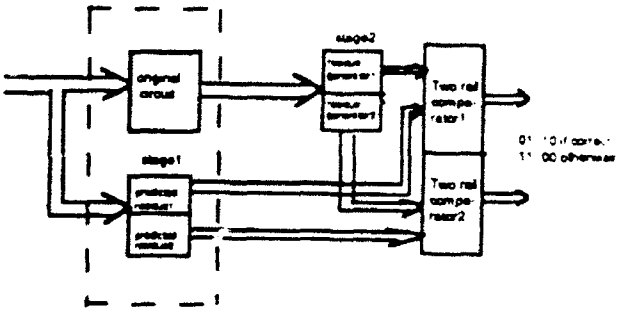

Fig 1. Block diagram of self-checking circuit

2. Concurrent error detection scheme

Before presenting the design steps, let us define the following:

Definition1: The *residue weight* of a bit in a output pattern is the residue of the binary weight of the bit mod m, where m is the selected check base. For example the residue weight of each bit in a 6-output pattern using mod7 check base are:

d5	d4	d3	d2	d1	d0
4	2	1	4	2	1

Definition2: Two errors are said to be *multiple exclusive* if they cannot occur in the same output pattern

To illustrate let us consider the output pattern given below

d5	d4	d3	d2	d1	d0
1	1	0	0	0	1

This output pattern can have a unidirectional error in bits (d0, d4) because bits d0 and d4 can change from 0 to 1, however it is not possible to have a bi-directional error in bits d0 and d4 in this pattern. Thus these two errors are mutually exclusive.

Definition3: If an output pattern is distributed into two groups the output bits present in both the groups are *common bits.*

Definition4: The output bits other than the common bits are called *unique bits.*

Definition5: Two bits are said to have unique weights if the residue weights assigned to them are not the same nor complementary.

The proposed scheme consists of the following steps:
(i) The output bits m(m=2n where n>2) of a circuit are distributed into two groups. Each group consists of (m-2) bits out of which (m-4) are common to both the groups. These (m-4) bits can be chosen randomly out of m output bits.For example, the output bits of a six output circuit (m=6) are distributed into two groups such that each group consists of 4(=m-2) bits out of which 2(=m-4) bits are common. One possible distribution is as shown below.

group2	group1
{d1 d0 d5 d4}	{d3 d2 d1 d0}

In this case, d0 and d1 are common bits, and (d2,d3) in group1 and (d4,d5) in group2 are unique bits.
(ii) Unique residue weights are assigned to the common bits of each group.
For example, if the common bits d0 and d1 are assigned residue weights 1 and 2 in group1, they should be assigned
different residue weights in the second group e.g. 2 and 1. The assignment is shown below.

group2	group1
{d1 d0 d5 d4}	{d3 d2 d1 d0}
{ 2 1 x x }	{ x x 1 2}

where x corresponds to residue weights yet to be assigned.

If any two common bits have same residue weights in one group, they will still have to be assigned unique residue weights in the other group. For example, if d0 and d1 have weights 1 and 1 in group1 they should be assigned unique residue weights in group2 e.g 4 and 2.

group2	group1
{ d1 d0 d5 d4}	{ d3 d2 d1 d0}
{ 2 4 x x }	{ x x 1 1}

(iii) If the residue of the sum of any three common bits is 0 in one group, they should be assigned residue weights such that the residue of their sum is not zero in the other group.

For example, in an 8 output circuit bits d0, d1 and d2 have residue weights 1, 2 and 4 in group1. If they are also present in group2 they can have residue weights of 4, 1 and 1 so that the residue of the sum of the residue weights is not equivalent to zero.

group2	group1
{ d3 d2 d1 d0 d7 d6}	{d5 d4 d3 d2 d1 d0}
{ x 1 1 4 x x }	{ x x x 4 2 1}

(iv)The two unique bits in each group are assigned unique residue weights.

In the example considered in step (ii) , (d5, d4) and (d3, d2) are unique bits in groups 1 and 2 respectively. These bits can be assigned residue weights of 1 and 4 respectively in both the groups. .

group2	group1
{ d1 d0 d5 d4}	{ d3 d2 d1 d0}
{ x x 1 4}	{ 1 4 x x }

The application of the above steps to a circuit with 6-outputs results a residue weight assignment as shown below

group2	group1
{ d1 d0 d5 d4}	{ d3 d2 d1 d0}
{ 1 2 1 4}	{ 1 4 2 1}

Note that unique weights are assigned to the common bits, d0 and d1, in each group. The unique bits in each group are also assigned unique weights. For example, d4 and d5 in group1 are assigned unique weights 1 and 4

respectively.

The following lemmas verify the error detection capability of self checking circuits designed using the above procedure.

Lemma1:

All single bit errors can be detected using this scheme.

Proof:

Since the change in residue due to the presence of a single bit error in either group is never 0, its detection can be guaranteed.

Lemma2:

All double bit errors, unidirectional and bidirectional, can be detected using this scheme.

Proof:

Since in mod7 a bit can have a residue weight of 1,2 or 4, a double bit unidirectional error will produce a non zero residue change e.g. 3,5,6. Hence, a double bit unidirectional error will always be detected. A double bit bidirectional error can be one of the following

i. both bits belong to one group only.
ii. one bit in each group.
iii. a common bit is erroneous.
iv. two common bits are erroneous, which will result in identical 2-bit error in each group.

case i : This error cannot be detected if and only if both bits have same residue weights. Since this is not possible, they will be detected.

case ii : By lemma1, this error combination is detectable.

case iii : Can be proved as in case (ii)

case iv : Can be proved as in case (i)

Thus all double bit bidirectional errors are detectable.

Lemma3:

All triple bit errors are detectable in circuits designed using the proposed scheme.

Proof:

A 3-bit error belongs to one of the following groups.

(i) two unique bits in one group and one bit in the other group.

(ii) two common bits and one unique bit.

(iii) one common bit and two unique bits.

(iv) three common bits.

casei : By step (iv) and lemma1 this error is detectable.

caseii : Two common bits will have unique weights in at least one group thus producing a residue different from the expected one.

caseiii : By lemma 1 this error is detectable.

caseiv : If the error is unidirectional it will be detected in at least one group [stepiii]. If it is bidirectional the only way it will remain undetected in one group if and only if at least two bits in that group have same residue weight, and also their sum is equivalent to the residue weight of one of the bit in that group.However, this error will be detected in the other group since the residue weights have been assigned in step(ii) in such a way that it will always result in a residue different from the expected one.

3.Application of the technique

Let us illustrate the application of the proposed scheme by considering 4-output and 6-output circuits.

i) Error detection in 4-output circuits:

The four output bits are divided into two groups as shown below

{d3, d2} {d1, d0}

Assignment of residue weights is done as follows.

{d3, d2}	{d1, d0}
{ 2, 1}	{ 2, 1}

An error will not be detected if there is no change in the residue because of the presence of error in the output bits is 0. Since the residues of each group are calculated and compared separately an error will go undetected if and only if both the groups fail to detect it. It can be easily

seen that it is not possible to get a residue change of 0 in the above shown grouping. Therefore all errors in the 4-output circuit will be detected i.e 100% error detection.

ii) **Error detection in 6-output circuit:**

As mentioned previously the 6-output circuit can be partitioned into two groups as shown below

group2	group1
{ d1 d0 d5 d4}	{ d3 d2 d1 d0}
{1 2 1 4}	{ 1 4 2 1}

The residue change due to the presence of an error in each group is calculated and compared with the residues that would have been obtained if there was no error in the circuit. The undetected error can be derived in the following manner.

a) First consider those errors in group1 which will not produce any change in the residue of group1, these are :
i) unidirectional error in bits d0,d1 and d2.
ii) unidirectional error in bits d1,d2 and d3.
iii) bi-directional error in bits d0 and d3.
iv) bi-directional error in bits d0,d3 and d1.
v) bi-directional error in bits d0,d1,d3 and d2.

b) Observe if the common bits are present in the errors. It can be noticed that at least one or both the common bits (d0,d1) appear in all the errors indicated above. Note the corresponding change in residue weights of these bits in group2.

c) The next step is to see whether change in the residue of the unique bits in group 2 is identical to the residue derived for for the common bits or its complement in group 2. If the is true, this multi bit error is undetectable.

To illustrate, let us consider a unidirectional error in the output bits d3,d2 and d1. There is no change in residue due to the presence of this error, which means that group1 will not detect this error. The change in the residue of the common bits in group2 is 1. Hence the error is detectable. However if unique bit d5 is also erroneous, then the resulting combination becomes undetectable. This is because the change in bit d5 will not result in any residue change in group2.

Let us now determine the probability of single and multibit errors being detected in a circuit designed using the proposed approach. The number of errors in an output pattern of a 6-output circuit are calculated as shown below.

Number of Single bit errors	=	6C1 = 6
Number of Double bit errors	=	6C2 = 15
Number of Triple bit errors	=	6C3 = 20
Number of Four bit errors	=	6C4 = 15
Number of Five bit errors	=	6C5 = 6
Number of six bit errors	=	6C6 = 1

Hence, the total number of errors is 63.

The following errors on 6-output pattern are undetectable:
i) bi-directional error in bits d0, d5, d4 and d3.
ii) bi-directional error in bits d1, d2, d3 and d5
iii) unidirectional error in bits d0, d1, d2, d5, d4.
iv) bi-directional error in d3, d0 and d1, d5.
v) bi-directional error in bits d0, d1, d3, d4 and d2.
vi) bi-directional error in bits d3, d1, d0, d5 and d2, d4.

Note that all the above mentioned errors except (iv) and (i, iii) are mutually exclusive to each other (by definition2). Hence, the maximum number of undetectable errors that can occur in a 6-bit output pattern is 2.

There are four output patterns in which the presence of errors (i, iii) or (i,ii) is undetectable. Thus, the percentage of detected errors in each of these 4 output patterns = 61/63 * 100 = 96.8%.

Let us consider next the effect of each of the individual errors (i-vi) on the output patterns. The number of output patterns in which only one of these errors can occur is derived in the following manner. There are three 4-bit errors, which can occur in two possible ways. For example, error (i) can be as follows.

d5	d4	d3	d2	d1	d0	->	d5	d4	d3	d2	d1	d0
1	1	0	x	x	1	->	0	0	1	x	x	1
0	0	1	x	x	0	->	1	1	0	x	x	0

There are two unassigned bits, which can be one of four possible combinations. Thus the number of output patterns in which error (i) remain undetected is 2**4 = 8. Since there are three such errors (i, ii, iv), the total number of output patterns in which these remain undetected = 3 * 8 = 24. These include the previously considered four output patterns in which (i, iii) or (i, ii) can occur. Thus the total number of output patterns which can have only a single error is 20 (=24 - 4). There are two 5-bit undetectable errors which can occur in two different ways. For example consider error (v)

d5	d4	d3	d2	d1	d0	->	d5	d4	d3	d2	d1	d0
x	1	1	0	1	1	->	x	0	0	1	0	0
x	0	0	1	0	0	->	x	1	1	0	1	1

There is one unassigned bit which can have two possible values. Thus, the number of output patterns in which error (v) remain undetected is 2* 4 = 8. These include two of the previously considered four patterns in which (i, iii) can occur. Thus the total number of patterns which can have only a single 5-bit error is 6(=8-2). There is only one six bit error which can occur in 2 output patterns.

Thus the total no of output patterns in which only one error can remain undetected = 20+6+2 = 28. In other words, the total number of detectable errors in these patterns = 63 -1 = 62. Hence the percentage of detectable errors =62/63 = .984 = 98.4%

In the remaining 32 (=64 -28 - 4) combinations all errors are detectable i.e percentage of detectable errors is 100%.

Therefore the average percentage of errors detected by using this method in 6-output circuit is
= (4*96.8 +28 * 98.4 +32 * 100) / 64 =99.1 %

Table1 shows the probability of detecting errors in 4, 6 and 8 output circuits.

No of outputs	Output Grouping	Mod value used	Probability of detection
4	{d3 d2} {d1 d0}	7	1
6	{d1 d0 d5 d4} {d3 d2 d1 d0}	7	0.991
8	{d3 d2 d1 d0 d7 d6} {d5 d4 d3 d2 d1 d0}	7	>0.984

TABLE 1.Error detection probability in multi output circuits.

4.Conclusion

A technique for designing circuits with concurrent checking capability has been proposed. About 99% of all the possible errors are detected using this scheme. Although the technique has been applied to circuits with upto 8-bit outputs, it can be extended to higher output circuits by taking higher check base, as long as the selected check base satisfies all the steps involved in grouping the output bits. The major advantage of the proposed technique is that the error detecting capability is based on the output bits of a circuit rather than its internal complexity. As long as a fault in a circuit corrupts an output pattern i.e produces single or multibit errors, the probability of its remaining undetected is very low.

Acknowledgement

This work was supported by the Air Force Office of Scientific Research under grant F49620-89-C-0069.

References

1. J. H. Lala and L. Alger "Hardware and software fault tolerance: a unified architectural approach" Proc. Symp on Fault Tolerant Computing, 1988, pp. 240-245.

2. T. R. Rao and H. Fujiwara "Error Control Coding for Computer Systems", Prentice Hall, 1989.

3. I. L. Sayers and D. J. Kinniment "Low-cost residue codes and their application to self-checking VLSI systems" IEE proc., Vol 132, No.4, July 1985, pp. 197-202.

4. P. K. Lala, F. Busaba, K.C. Yarlagadda "An approach for designing Self-checking logic using residue codes" IEEE VLSI Test Symposium 1991, pp. 166-171.

Appendix F

A STATE ASSIGNMENT TECHNIQUE FOR FULLY TESTABLE SEQUENTIAL CIRCUIT DESIGN

Xiaoying Ma and P. K. Lala
Electrical Engineering Department
North Carolina A&T State University
Greensboro, NC 27411, USA

ABSTRACT

A state assignment technique for designing testable sequential circuits is presented. All single stuck-at faults in a circuit including those affecting multiple signal lines, can be detected by the proposed technique.

1. Background and Definitions

A realistic approach to simplify testing of sequential circuits is to synthesize a testable circuit from the specification, which eliminates the post-design modification as in the scan-based technique. In this paper, a state assignment technique for implementing fully testable sequential circuits directly from their specifications e.g. state transition graphs (STGs) is presented.

The first step in the synthesis of a sequential circuit is to specify its behavior in terms of a STG (state transition graph), which represents the so-called 'specification machine'. The next step in the synthesis procedure is to determine the number of flip-flops needed to implement the machine i.e. the 'implementation machine'. One of the major problems in sequential circuit design is how to represent the symbolic states of the circuit with binary variables i.e. the state assignment. It has been shown that the choice of a proper state assignment can improve the testability of sequential circuits[1]. In recent years several logic synthesis procedures have been proposed which use special state assignment techniques to improve testability of sequential circuits [2,3]. However, none of them can achieve 100% testability for irredundant circuits, because detection of faults in the memory elements cannot be guaranteed. In addition, single stuck-at faults cannot be detected if such a fault occurs at a fan-out point, and affects more than one line in the circuit.

Before formally presenting the state assignment procedure, we define the following terms:

Definition 1: A **specification machine** is represented by the State Transition Graph (STG). An **implementation machine** is the physical realization of the specification machine; the number of states in the implementation machine is equal to or greater than those in the specification machine.

Definition 2: A **valid state** is a state which is specified in the STG. Any state which belongs to the implementation machine but not specified in the STG will be identified as **an invalid state.**

Definition 3: A next state corresponding to a present state is an **erroneous state** if it is not the same as the expected next state.

Definition 4: A state machine is **R-reachable** if for every state S_i in the STG, there exists an input sequence which will take the machine from the reset state to S_i.

Definition 5: A **single stuck-at fault** is a fault which causes only one node of a circuit to be held at a particular logic value and may drive more than one gate. If a single stuck-at fault only drives a single gate it will be called a **strictly single stuck-at fault.**

Definition 6: A fault is called **irredundant** if its presence causes circuit malfunction i.e. the fault is detectable. If a circuit has no undetectable faults, it is **irredundant.**

Definition 7: A fault is **on-line testable** if its presence can be detected during the normal operation of the circuit.

2. State Assignment Technique

For a reduced specification machine of P states, whose implementation version has Q states, the state assignment rules for fully testable design are:

1) all P (valid) states will have code words of distance-2 from each other, and those states that have same outputs will have distance-2k ($k > 1$) code words;

2) all Q - P (invalid) states will have different i.e. invalid outputs from that of all P states.

Theorem A sequential circuit implemented using the proposed rules is fully testable for all single stuck-at faults in output logic (OL), next state logic (NSL) and memory block (FF). The implementation machine is on-line testable for all single stuck-at faults in the next state logic, and at the outputs of memory elements which affect only next state logic.

For the sake of brevity we do not prove the theorem here.

2.3 CODES FOR STATE ASSIGNMENT

As indicated in the above theorem, any code with a Hamming distance of 2j (j = 1,2,...) can be used for representing the P valid states of a circuit. One such code

to the m-out-of-n code. An m-out-of-n code has m 1s and n - m 0s, with a Hamming distance of 2j (j = 1,2,..., $\lfloor n/2 \rfloor$) between the code words.

Next we consider how to select the m and n values for representing the states of a sequential circuit. Here n represents the number of flip-flops needed. In conventional design if the implementation machine corresponding to a specification machine of P states requires n-flip-flops, the total number of states $Q = 2^n$. However, if an m-out-of-n code is used, the total number of states $Q = n!/\{m!(n-m)!\}$. Here Q is the number of states in an implementation machine. So, to represent P states of a specification machine, the required number of flip-flops can be determined from the following inequality:

$$\frac{(n-1)!}{m!\,(n-m-1)!} < P \leq \frac{n!}{m!\,(n-m)!} \qquad (2.1)$$

While assigning code words to P valid states according to rule 1, those states that have identical output(s) should be assigned code words of distance 2d (d > 1). The invalid states (= Q - P) will have don't cares as next states, but their outputs should differ from that of all P valid states, i.e. they should be assigned a unique output value.

We select the m value for an m-out-of-n code such that whenever possible the number of code words will be sufficient to uniquely represent each valid state. In order to maximize the number of code words in m-out-of-n code, the best value of m is $\lfloor n/2 \rfloor$.

Table 2.1 shows the number of flip-flops required to represent the states for both conventional design and design for testability using k-out-of-2k code.

No. of states	No. of FFs (conventional)	No. of FFs (testable)
4	2	4
5-6	3	4
7-8	3	5
9-10	4	5
11-16	4	6
17-20	5	6
21-32	5	7
33-35	6	7
36-64	6	8
:		
127-128	7	10
129-252	8	10
253-256	8	11

Table 2.1. Number of flip-flops required for conventional and testable design

It can be seen that the number of additional flip-flops for testable design does not exceed 3 for testable implementation. Fig 1(a) shows the STG of an arbitrary sequential circuit. The state assig realization of the circuit is shown i shows the excitation and output testable circuit.

P.S.	x=0	x=1	y_4y (valid :
S_1	S_1, O_0	S_2, O_1	S_1 0
S_2	S_2, O_1	S_3, O_2	S_2 0
S_3	S_3, O_2	S_4, O_3	S_3 0
S_4	S_4, O_3	S_5, O_0	S_4 1 (
S_5	S_5, O_0	S_6, O_1	S_5 1
S_6	S_6, O_1	S_1, O_2	S_6 1 (
			(invalid state S_7 - S_{16}

(a) An arbitrary STG (b) State design

$$Y_1 = \bar{x}\, y_1 + x$$

$$Y_2 = \bar{x}\, y_2 + x y_3 \bar{y_2}$$

$$Y_3 = \bar{x}\, y_3 + x$$

$$Y_4 = \bar{x}\, y_4 + x y_4 \bar{y_2}$$

$$Z_1 = \bar{x}\, \bar{y_2}\, y_1 + \bar{x}\, y_4 \bar{y_3} + x$$

$$Z_3 = \bar{y_4}\, \bar{y_2}\, \bar{y_1} + \bar{y_4}\, \bar{y_3}\, \bar{y_2} + \bar{y_4}\, \bar{y}$$

$$Z_2 = \bar{x}\, y_4 y_1 + \bar{y_3} y_2 + x y$$

(c) Excitation and output eq

Fig 1. An example of the testable state

4.Conclusion

A state assignment technique for sequential circuits with partial on-line che been proposed. At most thre addition associated driving logic, and one extr needed to implement sequential circuits o of states.

5.References

1.S.Devadas et al "Optimal logic synthesis and testability:Two faces of the same coin" Proc. International Test Conference, pp.4-12, 1988

2.S.Devadas et al "Synthesis and optimization procedures for fully and easily testable sequential machines" Proc. International Conference, pp. 621-630, 1988

3.S.Devadas and K. Kreutzer "Boolean minimizaton and algebraic factorization procedures for fully testable sequential machines", IEEE International Conf. on CAD, pp.208-211, 1989.

Appendix G

Lemma 4:

Assume the number of errors which do not change the X-field is Q, and a subset of Q e.g. P is the number of errors which produce complementary residue values. If L is the number of unique residue weights for the check base used, then $Q \leq L+P$.

Proof:

The presence of an error pattern in the information bits may not change the X-field, but can be corrected if it changes the R-field such that the difference between the expected and actual residue value gives a unique value. If two such values are complement to each other, then the error patterns creating the change in the residue value have atleast one common bit. Suppose there are Q error patterns which do not change the X-field, and a subset P of Q produce complementary residue values. Assume for a chosen check base there are L unique residue weights. Thus (Q-P) will not exceed L, and for each pattern the difference between the expected and actual residue values is unique. In other words $Q-P \leq L$ or $Q \leq L+P$.

Lemma 5:

If the number of information bits are partitioned into 5.2^{i-4} groups then the inequality. $Q<L+P$ is satisfied.

Proof:

The number of errors which will not change the X-field can be obtained by considering the following cases:

case a: Two elements in a group:In a group with two elements if both bits are erroneous the relevant bit in the X-field remain unaffected. This corresponds to one error which will not change the X-field.

case b: Three elements in a group:

A group with three elements can have two 2-bit adjacent errors which will not change the X-field. Thus two errors will not change the X-field.

case c: Four elements in a group:

The bit in the X-field corresponding to a group with four elements, will not be affected if there are three 2-bit adjacent errors, or all four bits are erroneous. This corresponds to four errors which will not change the X-field

case d: Adjacent groups:

Two adjacent groups can have a four bit adjacent error (last two bits from one group and first two bits in its adjacent group) which will not change the X-field.

Table 4 shows the number of errors in various words of information bits which do not effect the X-field.

Information bits	**no of groups**	**No of errors, Q, not effecting the X-field**
8	3	7
16	5	15
32	10	33
64	20	67

Table 6: number of groups and the no of errors which do not effect X-field

After going through all the steps in the code construction, the number of possible errors producing complementary residue values for different information bits are found to be

Information bits	check base	No of errors, P, which produce complementary residues
8	13	2
16	29	4
32	61	9
64	125	17

Table 7 : information bits against no of complement errors

For all the information bits the inequality in lemma 4 is satisfied. Number of unique residue weights is got from table 1. The inequality is satisfied easily for information bits greater than 64 also, since the rate of increase of unique residue weights + complement errors is greater than that of no of errors not effecting X-field. A lesser value than the expression 5.2i-4 may also satisfy the expression but this value is selected so that it leaves some unique residue weights for the errors in the check bits to be corrected.

Example: Consider 32 bit information with 10 groups. 8-groups will have 3 bits each and 2 groups will have 4 bits each. Therefore the number of errors which will not effect the X-field is equal to 8*2(case b) + 2*4 (case c) + 9 (case d) (=33). This is lesser than the value: no of unique residues + no of complement errors, 30(table 4)+ 9(table 5) = 39.

Lemma 4:

Assume the number of errors which do not change the X-field is Q, and a subset of Q e.g. P is the number of errors which produce complementary residue values. If L is the number of unique residue weights for the check base used, then $Q \leq L+P$.

Proof:

The presence of an error pattern in the inforamtion bits may not change the X-field, but can be corrected if it changes the R-field such that the difference between the expected and actual residue value gives a unique value. If two such values are complement to each other, then the error patterns creating the change in the residue value have atleast one common bit. Suppose there are Q error patterns which do not change the X-field, and a subset P of Q produce complementary residue values. Assume for a chosen check base there are L unique residue weights. Thus (Q-P) will not exceed L, and for each pattern the difference between the expected and actual residue values is unique. In other words $Q-P \leq L$ or $Q \leq L+P$.

Lemma 5:

If the number of information bits are partitioned into 5.2^{i-4} groups then the inequality, $Q<L+P$ is satisfied.

Proof:

The number of errors which will not change the X-field can be obtained by considering the following cases:

case a: Two elements in a group:In a group with two elements if both bits are erroneous the relevant bit in the X-field remain unaffected. This corresponds to one error which will not change the X-field.

case b: Three elements in a group:

A group with three elements can have two 2-bit adjacent errors which will not change the X-field. Thus two errors will not change the X-field.

case c: Four elements in a group:

The bit in the X-field corresponding to a group with four elements, will not be affected if there are three 2-bit adjacent errors, or all four bits are erroneous. This corresponds to four errors which will not change the X-field

case d: Adjacent groups:

Two adjacent groups can have a four bit adjacent error (last two bits from one group and first two bits in its adjacent group) which will not change the X-field.

Table 4 shows the number of errors in various words of information bits which do not effect the X-field.

Information bits	**no of groups**	**No of errors, Q, not effecting the X-field**
8	3	7
16	5	15
32	10	33
64	20	67

Table 6: number of groups and the no of errors which do not effect X-field

After going through all the steps in the code construction, the number of possible errors producing complementary residue values for different information bits are found to be

Information bits	check base	No of errors, P, which produce complementary residues
8	13	2
16	29	4
32	61	9
64	125	17

Table 7 : information bits against no of complement errors

For all the information bits the inequality in lemma 4 is satisfied. Number of unique residue weights is got from table 1. The inequality is satisfied easily for information bits greater than 64 also, since the rate of increase of unique residue weights + complement errors is greater than that of no of errors not effecting X-field. A lesser value than the expression 5.2i-4 may also satisfy the expression but this value is selected so that it leaves some unique residue weights for the errors in the check bits to be corrected.

Example: Consider 32 bit information with 10 groups. 8-groups will have 3 bits each and 2 groups will have 4 bits each. Therefore the number of errors which will not effect the X-field is equal to 8*2(case b) + 2*4 (case c) + 9 (case d) (=33). This is lesser than the value: no of unique residues + no of complement errors, 30(table 4)+ 9(table 5) = 39.